



**ON THE USE OF SURROGATE FUNCTIONS
FOR MIXED VARIABLE OPTIMIZATION OF
SIMULATED SYSTEMS**

THESIS

John Elliott Dunlap, First Lieutenant, USAF

AFIT/GOR/ENS/05-06

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT/GOR/ENS/05-06

**ON THE USE OF SURROGATE FUNCTIONS FOR MIXED
VARIABLE OPTIMIZATION OF SIMULATED SYSTEMS**

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment for the Requirements for the
Degree of Master of Science in Operations Research

John Elliott Dunlap, B.S.

First Lieutenant, USAF

March 2005

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AFIT/GOR/ENS/05-06

**ON THE USE OF SURROGATE FUNCTIONS FOR MIXED
VARIABLE OPTIMIZATION OF SIMULATED SYSTEMS**

John Elliott Dunlap, B.S.

First Lieutenant, USAF

Approved:

Dr. James W. Chrissis
Thesis Advisor

Date

Lt Col Mark A. Abramson
Reader

Date

Abstract

This research considers the efficient numerical solution of linearly constrained mixed variable programming (MVP) problems, in which the objective function is a "black box" stochastic simulation, function evaluations may be computationally expensive, and derivative information is typically not available. MVP problems are those with a mixture of continuous, integer, and categorical variables, the latter of which may take on values only from a predefined list and may even be non-numeric. Mixed Variable Generalized Pattern Search with Ranking and Selection (MGPS-RS) is the only existing, provably convergent algorithm that can be applied to this class of problems. Present in this algorithm is an optional framework for constructing and managing less expensive surrogate functions as a means to reduce the number of true function evaluations that are required to find approximate solutions.

In this research, the NOMADm software package, an implementation of pattern search for deterministic MVP problems, is modified to incorporate a sequential selection with memory (SSM) ranking and selection procedure for handling stochastic problems. In doing so, the underlying algorithm is modified to make the application of surrogates more efficient. A second class of surrogates based on the Nadaraya-Watson kernel regression estimator is also added to the software. Preliminary computational testing of the modified software is performed to characterize the relative efficiency of selected surrogate functions for mixed variable optimization in simulated systems

Acknowledgments

I would like to express my sincere appreciation and gratitude to my faculty advisor, Dr. James W. Chrissis, for his generous guidance, patience, and support throughout the course of this thesis effort. His timely insights and gentle nudges made this work manageable. I would also like to thank my reader, Lt Col Mark A. Abramson, for sparking my interest in pattern search methods, guiding me to its correct application for this research, and molding this document to adequately express my thoughts. As a direct result of your efforts, I am now proud of this final product. Additionally, I would like to thank Maj Todd A. Sriver upon whose work a majority of this thesis is based and for leading me toward some of the more important issues.

Most importantly, I want to say a special thank you to my family for all of their love and sacrifice. First to my wife, whose support and encouragement have allowed me to succeed. Her persistence and guidance during this endeavor enabled me to focus on the task at hand while never forgetting what is really important in life. Finally to my children, for their patience and understanding when I had to "go to school" while remaining at home during all of those evenings and weekends, thank you.

John E. Dunlap

Table of Contents

	Page
Abstract	iv
Acknowledgments	v
List of Figures	x
Chapter 1. INTRODUCTION	1
1.1 Problem Setting	1
1.2 Approach	8
1.2.1 Use of Stochastic Simulation Models	8
1.2.2 Generalizing the Algorithm	9
1.3 Summary	10
1.4 Overview	11
Chapter 2. LITERATURE REVIEW	12
2.1 Unconstrained Optimization Algorithms	12
2.2 Newton-based Optimization Algorithms	13
2.2.1 Guaranteeing Convergence	14
2.2.2 Suitability of Derivative-based Methods	17
2.3 Search Heuristics	18
2.3.1 Simulated Annealing	18
2.3.2 Evolutionary Algorithms	21
2.3.2.1 Genetic Algorithms	22

	Page
2.3.2.2 Evolution Strategies	23
2.3.3 Suitability of Search Heuristics	25
2.4 Generalized Pattern Search (GPS) Methods	26
2.4.1 Pattern Search Methods	26
2.4.2 Extensions to Constrained Problems	29
2.4.3 General Pattern Search Extensions	34
2.4.4 Suitability of Generalized Pattern Search (GPS) Methods	37
2.5 Implementation Considerations	38
2.6 Summary	45
Chapter 3. APPROACH	46
3.1 Mesh and Poll Set Construction	46
3.2 Optimality for Mixed Variable Domains	49
3.3 Bound and Linear Constraints	51
3.4 MGPS Algorithm	53
3.5 Proposed Modifications	56
3.5.1 Search Approach	57
3.5.2 R&S Procedure	59
3.6 Implementation Considerations	66
3.7 Summary	67
Chapter 4. COMPUTATIONAL EVALUATION	68

	Page
4.1 Research Surrogates	68
4.1.1 Kernel Regression	69
4.1.2 Kriging	71
4.2 Surrogate Sampling Design	73
4.2.1 Latin Hypercube Sampling	73
4.2.2 The Merit Function	74
4.2.3 The Trust Region	75
4.3 Implementation Considerations	76
4.4 Design of Investigation	78
4.4.1 Deterministic Runs	80
4.4.2 Pilot Study	80
4.4.3 Main Runs	83
4.5 Computational Results	83
4.5.1 Main Run Results	84
4.5.2 R&S Modification Results	86
4.6 Summary	88
Chapter 5. CONCLUSIONS AND RECOMMENDATIONS	89
5.1 Future Research	89
5.1.1 Nonlinear Constraints	89
5.1.2 R&S Modifications	91

	Page
5.1.3 Surrogate Modifications	91
5.2 Summary	92
Appendix A. SUPPORTING DATA SUMMARY	93
A.1 Deterministic Run Data Summary	93
A.2 Pilot Study Data Summary	97
A.3 Main Run Data Summary	107
A.4 R&S Procedure Adjustment Results	117
A.5 Modified Main Run Data Summary	120
Bibliography	123
Index	128

List of Figures

Figure	Page
1.1	Simulation-Based Optimization Method 2
2.1	General Optimization Algorithm (adapted from Nash and Sofer [43]) 13
2.2	Backtracking Line Search (adapted from Dennis and Schnabel [19]) 17
2.3	General Simulated Annealing Algorithm 20
2.4	General Genetic Algorithm (adapted from Spall [54]) 24
2.5	General Evolution Strategies Algorithm (adapted from Spall [54]) 25
2.6	General Pattern Search Algorithm (adapted from Audet and Dennis [6]) 32
2.7	Triangular Continuation Region 43
3.1	Tangent Cone Near Boundary (adapted from Lewis and Torczon [36]) 52
3.2	MGPS Algorithm for Deterministic Optimization (adapted from Abramson [2]) 56
3.3	Triangular Continuation Region for SSM 60
3.4	Sequential Selection with Memory (adapted from Pichitlamken and Nelson [48]) 63
3.5	Modified MGPS Algorithm for Simulation Optimization 65
4.1	Examples of Latin Hypercube Samples of Strengths 1 and 2. 74

ON THE USE OF SURROGATE FUNCTIONS FOR MIXED VARIABLE OPTIMIZATION OF SIMULATED SYSTEMS

Chapter 1 - Introduction

1.1 Problem Setting

Consider the problem of optimizing a real system in which the objective is to find a set of controllable parameters that minimizes some measure of performance. To validate the results, direct experimentation on the system is preferred, but is not always possible, due to technological or cost restrictions. In this case, a common practice is to create a mathematical model of the system in terms of logical and quantitative relationships [33]. For a given performance measure with a defined level of abstraction, a valid mathematical model can usually be derived which accurately represents the behavior of the actual system. If the mathematical model is simple enough, then an analytic optimal solution may exist in closed form. However, for more complex models, simulated mathematical models (often referred to as *simulation models*) can produce representative outputs of the real system. The focus of this research is on simulation-based models.

Due in part to the ease of changing simulation configurations, simulation models have become popular in performing such tasks as characterizing system behavior, aiding in the design of real systems, and examining alternatives through what-if scenarios to improve the related system performance. For example, simulation models are routinely constructed to determine costs associated with manufacturing, warehousing, ordering, and shipping of goods under different production disciplines. If the system constraints and performance measure(s) of interest are modeled in the simulation, then the simulation responses can be used with an optimization algorithm to determine production strategies that optimize a performance measure for the underlying system. The resulting model solution can thus

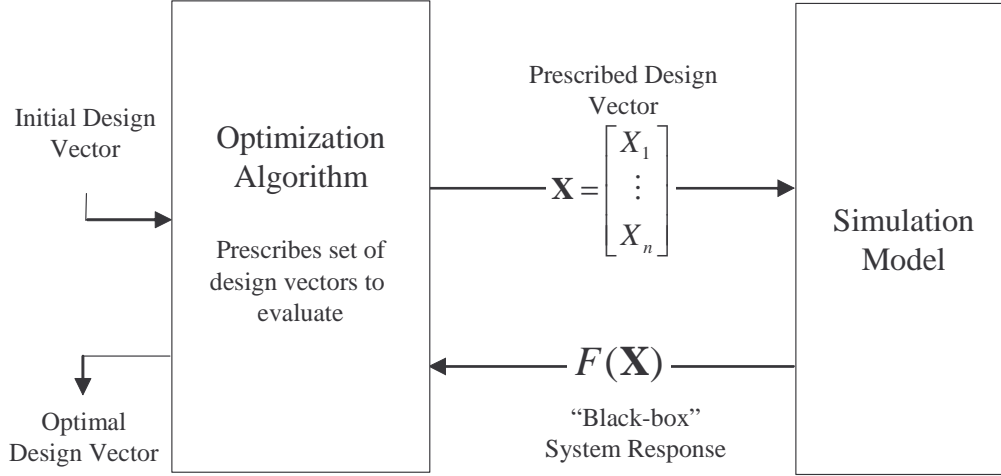


Figure 1.1. Simulation-Based Optimization Method

assist decision makers in developing proper courses of action by augmenting their knowledge of the actual system.

The term *simulation-based optimization* describes methodologies where complex systems are "designed, analyzed, and controlled by optimizing the results of computer simulations" (Kolda *et al.* [31]). As illustrated in Figure 1.1, the complex mathematical model representing the system is simulated (usually repeatedly) at a design point specified by an optimization algorithm, generating a response that is an estimate of the system's performance measure. In this context, simulation is defined as a numerical procedure that takes a design vector of inputs and generates a response based on underlying model relationships. For a given level of abstraction, a valid model can represent the system's set of controllable parameters as an input design vector. Using the abstracted design vector as input, the model generates an estimate of the system's performance measure. The resulting model response is then used by an optimization algorithm to introduce a new set of input design points to be evaluated by the model. The optimization algorithm thus acts on the model, rather than on the actual system, in order to optimize the performance measure. Under

the assumption of model validity, the set of controllable system parameters that optimize actual system performance will directly correspond to the resulting optimal design vector.

Although both cases are applicable to this research, it is important to note that these simulation models can be classified into two groups, depending on the presence of random (or probabilistic) components in the model. Simulation models that do not contain random components are called *deterministic models*. Deterministic models do not include random components; they explicitly represent the underlying system, usually by the use of schedules or differential equations. Although these models are generally analytically intractable, the model needs to be run only once at a particular design vector to produce a unique response of the associated system performance.

Simulation models which contain random components are called *stochastic models*. When modeling the actual system of interest, random components are often used to represent the uncertainty of the input parameters in the model. Although these simulation models are better able to account for the uncertainty, random components produce random noise in the evaluation of the response function; thus, the outputs from stochastic models are often referred to as *random responses*. To reduce the error in estimating the true response, repeated sampling of a particular design vector is traditionally performed to generate a distribution of the associated system performance. Although a response distribution can be generated from repeated sampling, in this research the simulation model will be assumed to have a terminating condition that produces a single response. Therefore, approaches used in non-terminating simulation models to estimate a performance measure, such as replication/deletion and batching means (see Law and Kelton [33]), are not required.

Random noise in the evaluation of the responses from stochastic simulation models requires special attention when estimating the performance measure. In the case of de-

terministic models, the simulation response $F(x)$ is a noise-free estimator of the system performance $f(x)$, which is dependent only upon the design vector x ; thus expected performance is given by

$$f(x) = \mathbb{E}[F(x)] \quad (1.1)$$

where $x \in \mathbb{R}^n$ is the design vector. In the case of stochastic models, the response of the simulation $F(x, \psi)$ depends not only on the design vector x , but also on the random noise of the system ψ ; thus expected performance is given by

$$f(x) = \mathbb{E}[F(x, \psi)] = \int_{\Omega} F(x, \psi) P(d\psi) \quad (1.2)$$

where x is the design vector, $\psi \in \Psi$ is an element of an underlying probability space (Ψ, \mathcal{F}, P) with sample space Ψ , sigma-field \mathcal{F} , and probability measure P . Because the responses are generated from a “black box” system and the simulations are analytically intractable, the probability distribution that defines $F(x, \psi)$ is assumed to be unknown but can be sampled. In both deterministic and stochastic models, the simulation responses $F(\cdot)$ provide an estimate of the system performance $f(x)$. Therefore, the *objective function* f of the underlying system is optimized by systematically improving the simulation response function F .

In order to optimize the underlying system, the simulation response can be systematically improved by the use of an appropriate optimization algorithm. However, the choice of an algorithm is restricted by certain assumptions. In this research, the simulation response is assumed to originate from a “black box”; thus analytical derivatives are typically not available. Furthermore, the responses may have few correct digits; therefore, approximation of the gradient by techniques such as finite differencing or simultaneous perturbation (see Spall [54]) may be unreliable. The applicable methods for this research can be

further restricted by allowing objective or constraint functions to be discontinuous, have undefined regions, or fail to return a solution at certain design points. Because of these considerations, gradient-based optimization algorithms cannot be effectively applied.

The selection of an applicable optimization algorithm is also restricted by the presence of *categorical variables*, which are those that can only take on values from a predefined list, and may have no ordinal relationship to one another. This restriction is common in the structure of many real-life systems. For example, a structural design problem may include the type of material as a variable (*e.g.* steel, aluminum, etc.), a production system may use different processing disciplines for a particular machine (first-in-first-out, last-in-first-out, priority, etc.), or a networking problem may be constructed with decision variables to indicate if a particular supply point is present (yes, no). These categorical variables can be represented by integers, but the values usually have no inherent ordering (*e.g.* 1 = steel, 2 = aluminum, *etc.* for the types of material). A further complication is that changes to categorical variable values may change the constraints of the problem and even the number of continuous variables. The class of optimization problems that includes continuous, integer-valued, and categorical variables is known as *mixed variable programming* (MVP) (Audet and Dennis [5]).

MVP problems with only a few categorical variables that can assume only a small number of values are usually not difficult to solve. In these problems, categorical settings can be exhaustively enumerated to produce subproblems which are solved and compared to determine the optimal setting. However, real-life systems may involve a large number of discrete variable combinations. Since repeated simulation model evaluations would be required to solve each of the subproblems at each categorical setting combination, this approach may be too computationally expensive to use as a solution approach. Because the

values of the categorical variables are assigned from an unordered, predefined list, relaxation techniques used for traditional integer programming problems (such as branch and bound) are not applicable. Relaxation techniques are able to avoid exhaustive enumeration in integer programming problems by exploiting the ordinality of the integer-valued variables; but in MVP problems, the categorical variables must be assigned a value from the list at every iteration, not just at the solution.

This research considers the optimization of simulation problems with mixed variables, where the continuous variables are bounded by linear constraints. This class of MVP problems can be expressed as

$$\min_{x \in \Omega} f(x) \quad (1.3)$$

where $f : \Omega \rightarrow \mathbb{R} \cup \{+\infty\}$ is a function of unknown analytical form (referred to as the *objective function*) and $x \in \Omega$ is the vector of mixed design variables. The mixed variable domain Ω is partitioned into continuous and discrete domains Ω^c and Ω^d , respectively, where some or all of the discrete variables may be categorical. Thus, each vector $x \in \Omega$ can be denoted as $x = (x^c, x^d)$ where $x^c \in \mathbb{R}^{n^c}$ are the continuous variables of dimension n^c and $x^d \in \mathbb{Z}^{n^d}$ are the discrete variables of dimension n^d . The domain of the discrete variables $\Omega^d \subseteq \mathbb{Z}^{n^d}$ can be represented as a subset of \mathbb{Z}^{n^d} by mapping the predefined list elements for each categorical variable to integer values. The domain of the continuous variables $\Omega^c \subseteq \mathbb{R}^{n^c}$ is restricted by linear constraints which can be functions of x^d . Since the value of the constraint bounds may be dependent on the value of the discrete variables, for each fixed set of discrete variable values x^d the associated constraints can be expressed as

$$\Omega^c(x^d) = \{x^c \in \mathbb{R}^{n^c} : \ell(x^d) \leq A(x^d)x^c \leq u(x^d)\} \quad (1.4)$$

where $A(x^d) \in \mathbb{R}^{m^c \times n^c}$, $\ell(x^d), u(x^d) \in (\mathbb{R} \cup \{\pm\infty\})^{m^c}$, $\ell(x^d) \leq u(x^d)$, and $m^c < \infty$. For convenience, in the remainder of this paper, the explicit dependence of Ω^c on x^d as shown in Equation (1.4) is omitted.

Due to this inherent complexity, few methods can efficiently solve MVP problems. One method that can provide proven convergence to a stationary point for linearly constrained and continuously differentiable MVP problems is the Audet-Dennis Mixed Variable Generalized Pattern Search (MGPS) [5], which is described in Section 2.4.1. This algorithm is implemented in the NOMADm MATLAB[®] software [1]. NOMADm is unique in that it not only provides an implementation of MGPS, but it also handles nonlinear constraints through a filter approach for infeasible iterates (see Audet and Dennis [8]). Since NOMADm is only applied in this research to problems with linear constraints, the filter approach is avoided.

Included in the MGPS algorithm is an optional SEARCH step at each iteration, in which the user may attempt any finite strategy to reduce computational cost. As noted by Booker *et al.* [10], inexpensive surrogate functions can be used within the SEARCH step to accelerate convergence to a solution without affecting the overall convergence theory. If the surrogate is reasonably accurate, then the search may advance the optimization algorithm to good solutions with fewer function evaluations than without a SEARCH step. Since improvement can be gained through inexpensive surrogates, Sriver [55] used a surrogate-based search in a computational evaluation of the underlying pattern search ranking and selection algorithm used in this research. Sriver [55] also suggested that future work include the examination of surrogate families for stochastic pattern search methods, which this research does in the context of stochastic MVP problems.

1.2 Approach

This research focuses on improving the MGPS algorithm for MVP problems used in NOMADm by:

1. Allowing for the use of "black box" simulation models with stochastic responses, and
2. Generalizing the MGPS algorithm for improved use of surrogates.

Allowing for stochastic simulation responses enables NOMADm to become applicable to a wider set of problem types. Generalizing the algorithm can reduce the number of function evaluations required for convergence to a stationary point; therefore, computational efficiency may be improved.

1.2.1 Use of Stochastic Simulation Models

As part of the MGPS algorithm, the incumbent best solution is only updated when a trial point provides a better response. When deterministic simulation models are used, this ensures monotonic system performance improvement, since selection of the incumbent is accomplished by comparison of deterministic system responses. However, when stochastic simulation models are used, the randomness of system responses complicates the selection process. To achieve a given statistical level of confidence that the selected iterate provides true system performance improvement, multiple replications may be required to obtain enough observations to be confident that the incumbent selected is the true "best" iterate (see Pichitlamken and Nelson [48]). Since replications require additional function evaluations, stochastic simulation models are often expensive to optimize. Using the simple statistical technique of pairwise comparison, the number of function evaluations required to solve the iterate selection subproblem has been previously considered too computationally inefficient to implement in the current optimization algorithm (see Trosset [60]). Recent developments in ranking and selection (R&S) statistical methods offer the ability to use

multiple comparisons to solve the iterate selection subproblem in a more efficient manner than pairwise comparison. Srivier demonstrated in [56] that the use of ranking and selection procedures for the iterate selection subproblem can provide an efficient selection of the incumbent while maintaining the required level of statistical significance. Thus, an R&S procedure is used to control the selection of the incumbent.

1.2.2 Generalizing the Algorithm

As noted in the previous subsection, the current MGPS algorithm is able to provide monotonic improvement by allowing an incumbent to be replaced only by an iterate which provides a better response. Response improvement alone, however, does not guarantee convergence to a stationary point for continuously differentiable functions with linear constraints. The MGPS algorithm further restricts the candidate iterates to meet polling conditions, described in Section 2.4.1, in order to meet underlying convergence theory. The MGPS algorithm evaluates points on a mesh and includes an optional SEARCH step where "a finite search, free of any other rules imposed by the algorithm, is performed anywhere on the mesh" (Audet and Dennis [5]), which is described in Section 3.1. The flexibility of the SEARCH step allows the user to employ any strategy favorable to the user, as long as it searches finitely many points (including none). Since the SEARCH step is optional, the user can fully personalize the SEARCH step with the understanding that underlying convergence theory is provided by the other steps of the algorithm.

Not only does the current MGPS algorithm allow for only one SEARCH step per iteration, but the search is restricted to the discrete plane (*i.e.*, same categorical variable settings) of the incumbent. This research explores the use of an additional SEARCH step that allow the user the flexibility to apply a finite search on the continuous regions associated with any choice of the discrete variable values. This additional SEARCH step enables

a more global search of the feasible region that may result in faster convergence to a stationary point or an improved solution at the termination of the algorithm. For problems with a small number of discrete planes, the additional SEARCH step may coincide with the current SEARCH step and thus may not benefit the algorithm significantly. However, for problems with a large number of categorical setting combinations, the additional SEARCH step enables the algorithm to update the incumbent with any iterate that provides a better response before polling is performed. Therefore, the reduction in function evaluations may be more significant in problems with a large number of categorical variables.

1.3 Summary

Because of the generality of the targeted class of optimization problems and the relatively weak assumptions, very few methods are provably convergent for MVP problems. A derivative-free approach, which has been shown to be convergent for MVP problems, is the mixed variable generalized pattern search (MGPS) algorithm. The purpose of this research is to develop an algorithm that can be used to efficiently solve simulation-based optimization problems. The research goal is to improve the efficiency of the MGPS algorithm proposed by Sriver, extend the applicability of NOMADm to stochastic, as well as deterministic, simulations, and perform testing of appropriate surrogates. Specifically, the ranking and selection method is used to control the iterate selection subproblem within NOMADm, which when combined with an additional SEARCH step and surrogate functions, increases the applicability and computational efficiency of NOMADm. This approach is applied to stochastic and deterministic, continuous and mixed variable programming problems, and performance is compared to that of the native NOMADm performance in terms of the quality of the terminating solution and computational efficiency.

1.4 Overview

The next chapter describes pertinent background information found in the literature. Chapter 3 describes the specific ranking and selection procedure as well as the surrogate functions used in this research and how they were integrated into NOMADm. Chapter 4 gives comparative results of this implementation against the algorithm currently used in NOMADm as applied to nonlinear programming problems described in Equation (1.3). Finally, Chapter 5 gives conclusions and recommendations for areas of further research.

Chapter 2 - Literature Review

The class of problems that can be represented as a simulation-based MVP structure is so general that a controlled approach must be used in developing a provably convergent, computationally efficient optimization algorithm. Before addressing modifications to the current algorithms, a review of the development and characteristics of the methods used to solve related problems is required. In this chapter, the general optimization algorithm is shown to be tailorable to the availability of accurate problem information, with the frequently used Newton-based methods discussed as an example. Direct search methods are then discussed with regard to their applicability to this research. Finally, an overview of ranking and selection procedures and surrogates are discussed.

2.1 Unconstrained Optimization Algorithms

Consider the problem of minimizing an unconstrained continuously differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\min_x f(x) \tag{2.1}$$

The selection of an algorithm for solving the problem in Equation (2.1) is usually dependent on the relative importance of computational efficiency and proven convergence to a stationary point of f .

A basic optimization algorithm for solving the problem in Equation (2.1) is given in Figure 2.1. By simply requiring that the objective function decreases during each iteration, the algorithm can produce a sequence of iterates that improves the objective function value. If derivative information is available, a common practice is the tailoring of the basic optimization algorithm into a Newton-based method. By placing restrictions on the search direction p_k and step size a_k at each iteration, Newton-based methods can,

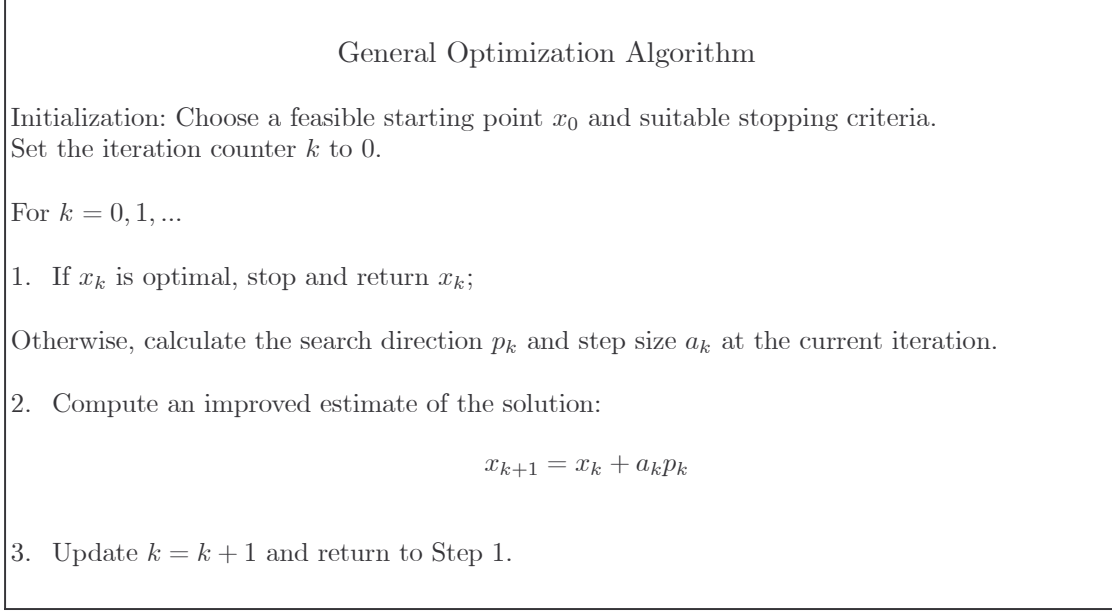


Figure 2.1. General Optimization Algorithm (adapted from Nash and Sofer [43])

under mild conditions, provide computationally efficient, proven convergence to a stationary point. Since Newton-based methods are so frequently used and provide the main concepts for proven convergence, the following section discusses the basic optimization algorithm in terms of a Newton-based algorithm. However, it should be noted that the Newton-based optimization algorithms are only one example of the general optimization algorithm in Figure 2.1.

2.2 *Newton-based Optimization Algorithms*

Under ideal conditions, Newton's method provides proven convergence to a minimizer at a quadratic rate, which, roughly speaking, means that the number of correct digits doubles at each iteration (Nash and Sofer [43]). For Newton's method, the search direction p_k is calculated as the solution of the *Newton Equation*

$$[\nabla^2 f(x_k)] p_k = -\nabla f(x_k) \tag{2.2}$$

Unfortunately, the cost of calculating both the gradient $\nabla f(x_k)$ and Hessian $\nabla^2 f(x_k)$ at each iteration can be computationally expensive, and the Hessian can become ill-conditioned (Nocedal and Wright [46]). Numerous methods have been developed which attempt to preserve the convergence rate while addressing these issues by replacing the Hessian with an inexpensive approximation. For quasi-Newton methods, the Hessian is replaced by a positive definite approximation B_k that is obtained and updated at a lower cost. Usually, the update methods that are used to calculate B_k also have conditions to prevent the approximated Hessian from becoming too ill-conditioned (for more details, see Nocedal and Wright [46]). For the steepest descent method, the Hessian is replaced by the identity matrix so that $p_k = -\nabla f(x_k)$. Although the steepest descent method is able to yield computational savings in the calculation of the search direction, convergence may become arbitrarily slow (Nash and Sofer [43]).

2.2.1 *Guaranteeing Convergence*

Although the general algorithm in Figure 2.1 includes those that converge to a stationary point, an unmodified algorithm may not make sufficient progress towards the point and, in some cases, may actually diverge. In an effort to enforce improvement in the solution estimate at each iteration, the step size a_k is computed in the direction p_k so that *simple decrease*, $f(x_{k+1}) < f(x_k)$, in the objective is achieved. That is, since $x_{k+1} = x_k + a_k p_k$, the simple decrease condition can be written as

$$f(x_k + a_k p_k) < f(x_k) \tag{2.3}$$

Under mild assumptions, requiring improvement at each iteration will keep the algorithm from diverging, but does not ensure convergence to a stationary point. Dennis and Schnabel [19, p. 117] provide the following example to illustrate that an algorithm relying solely on

a simple decrease condition may not converge to a stationary point of the objective. The function $f(x) = x^2$ has only one stationary point at $x = 0$. If the step sizes are chosen too small $a_k = 2^{-k+1}$, then the sequence of iterates is given by $x_k = 1 + 2^{-k}$, which converges to 1. If the step sizes are chosen too large $a_k = 2 + 3(2^{-(k+1)})$, then the sequence is given by $x_k = (-1)^k(1 + 2^{-k})$, with limit points at ± 1 . The choice of search direction can also affect the convergence of the algorithm. If the search direction p_k is almost orthogonal to the direction of steepest descent, the algorithm can also stall (Kolda *et al.* [31]) at a non-stationary point. Therefore, additional conditions must be met for optimization algorithms to prove convergence for unconstrained optimization problems.

Optimization algorithms generally use either an embedded trust region or a line search methodology to meet conditions that prove convergence to a stationary point of the objective. Since this review is used to support the convergence theory presented later in the document, only the line search conditions are presented. For line search methods, conditions are imposed on both the search directions and the step sizes in order to satisfy convergence conditions. The first condition required of a search direction is that the candidate direction must be a *descent direction*. A sufficient condition for p_k to be a descent direction is that $p_k^T \nabla f(x_k) < 0$. This condition ensures that the simple decrease condition in Equation (2.3) will be satisfied for sufficiently small a_k . The only other condition imposed on the search directions is the *sufficient descent condition* (Nash and Sofer [43, p. 314]), given by

$$\frac{-p_k^T \nabla f(x_k)}{\|p_k\| \|\nabla f(x_k)\|} \geq \varepsilon > 0, \quad k = 0, 1, \dots \quad (2.4)$$

for some $\varepsilon > 0$. This keeps the search direction p_k from approaching orthogonality to $-\nabla f(x_k)$ in the limit by bounding the angle between the two. This is more intuitively

seen if the sufficient descent condition is rewritten as $\cos \theta \geq \varepsilon > 0$, where θ is the angle between p_k and $-\nabla f(x_k)$, often referred to as the *angle condition* (Nash and Sofer [43, p. 314]).

Given these two conditions on p_k , step sizes must also be controlled to ensure convergence to a stationary point. The *sufficient decrease condition*, given by

$$f(x_k + a_k p_k) \leq f(x_k) + \mu a_k p_k^T \nabla f(x_k) \quad (2.5)$$

where $\mu \in (0, 1)$, ensures that the step size a_k produces a decrease that is at least as good as a fraction of what the linear approximation to $f(x_k + a_k p_k)$ would produce. Since the linear approximation of $f(x_k + a p_k)$ is given by

$$f(x_k + a p_k) \approx f(x_k) + a p_k^T \nabla f(x_k) \quad (2.6)$$

and the descent direction necessarily satisfies $p_k^T \nabla f(x_k) < 0$, the use of the sufficient decrease condition eliminates the possible selection of step sizes a_k that are too long. In order to avoid step sizes that are too short, a *curvature condition* must be satisfied. One such condition, is given by

$$\nabla f(x_k + a_k p_k)^T p_k \geq \eta \nabla f(x_k)^T p_k \quad (2.7)$$

where η is a scalar and $0 < \mu < \eta < 1$. This ensures that the slope at a step size a_k is at least η times the slope at the current point. The curvature condition eliminates potential choices of the step size which would produce slopes that are arbitrarily close to the gradient at the current point, thereby ensuring the step size a_k is not too short. Equation (2.5)

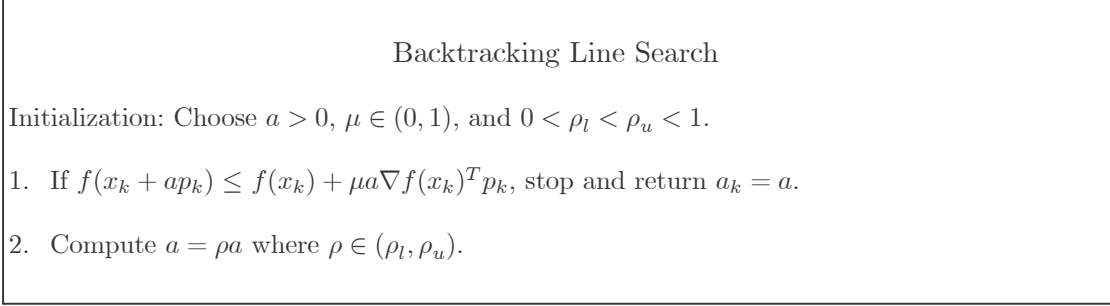


Figure 2.2. Backtracking Line Search (adapted from Dennis and Schnabel [19])

and (2.7) together are called the Wolfe conditions. Alternative conditions are discussed in Nocedal and Wright [46].

A simple and convenient line search strategy is called *backtracking*, which is detailed in Figure 2.2. In backtracking, the step size is contracted iteratively until the sufficient decrease condition is satisfied. Since p_k is a descent direction, backtracking terminates in a finite number of steps. Backtracking also enforces the curvature condition automatically (for more details, see Nocedal and Wright [46]).

2.2.2 Suitability of Derivative-based Methods

Although under ideal conditions methods that use derivative information (*i.e.* Newton-based methods) can provide quadratic rates of convergence, the assumption that first derivative information is available (or can be accurately calculated) is not satisfied for all cases of unconstrained optimization problems. In particular, when complex simulations are used to provide function evaluations, it may be difficult or impossible to calculate, or even estimate, the derivatives. For example, explicit derivative formulas may not exist for deterministic simulations that use numerical computation of differential equations to determine function evaluations. Additionally, approximation methods used on stochastic simulations, such as finite-differencing or simultaneous perturbation stochastic approximation (Spall [54]), may fail when function values are either only accurate to a few digits or computationally prohibitive in higher dimensions. Therefore, methods which rely on the availability of deriv-

ative information (and clearly those which depend on exact or approximate second-order derivative information, such as quasi-Newton and Levenberg-Marquardt methods) may not be rigorously applicable when developing an algorithm to solve simulation optimization problems.

2.3 Search Heuristics

Direct search methods use direct comparison of functional responses to progress to a point that provides an improved response. Because they do not use derivative information, direct search methods make fewer assumptions of the response generator (simulation model). Since direct search methods only require function responses, they are generally more broadly applicable than Newton-based methods. However, in exchange for generality, direct search methods may require a large number of function evaluations in order to converge to a solution, assuming that they converge at all.

As a member of the direct search methods class, search heuristics seek an acceptable improvement rather than a provably optimal solution by methodically searching the feasible region. Because of their simplicity and ability to discover good solutions rapidly (even in high dimensional settings), search heuristics have become popular in solving complex problems in a variety of different disciplines. As representatives of search heuristics, both simulated annealing and evolutionary algorithms are described in general and as they relate to this research. Each is an adaptation of a multidisciplinary technique originating from an engineering discipline.

2.3.1 Simulated Annealing

For simulated annealing, thermodynamic equilibrium theory is applied to the selection of iterates which do not satisfy the simple decrease condition of Equation (2.3). As shown in Figure 2.3, incumbent replacement is based on the objective function evaluation of the

incumbent and trial points. Trial points that provide simple decrease always replace the incumbent; however, trial points that fail to satisfy this condition may also replace the incumbent according to a probability distribution that is maintained by the algorithm. The probabilistic selection offered by simulated annealing theoretically allows the search algorithm to escape areas of local optima and actively search for a global optimum.

Annealing is the process of heating and slowly cooling an object (typically glass or metal) according to a schedule in order to strengthen or harden the resulting product. For optimization of systems, the object of interest is represented by the solution where strength is measured as proximity to optimality. For complex optimization problems, such as steel production or crystal formation, the *cooling schedule* directly affects the quality and cost of the resulting product. For example, in the formation of crystal from a liquid, the cooling schedule controls the resulting arrangement of atoms in the resulting solid crystal. If the transition of physical states occurs too rapidly, the atoms will be unable to adequately explore local rearrangements resulting in the atoms being trapped in the configuration they had in the liquid state. Thus, the slower the cooling schedule, the more likely that the atoms will find the ordering that has the lowest global energy. On the other hand, if the system is cooled too slowly, the cost to maintain the temperature offsets the relative value of the resulting crystal.

The concept of annealing was applied to numerical methods to minimize the resulting energy of a system by including thermodynamic fluctuations in statistical mechanical systems. Metropolis *et al.* [41] proposed the use of a state probability distribution to simulate a system at a fixed temperature. The Boltzmann-Gibbs energy state probability distribution provides the relative probability of seeing a system in a state with an energy E as:

General Simulated Annealing Algorithm

Initialization: Choose a feasible starting point x_0 and T_{\min} .
Initialize the cooling schedule used to generate the temperature T .
Set $x = x_0$.

1. If $T > T_{\min}$, stop and return x .
2. Randomly select a neighbor point y .
3. If $(\mathbb{E}[y] < \mathbb{E}[x])$, set $x = y$;
Otherwise, set $x = y$ with probability $\exp(-\frac{\mathbb{E}[y] - \mathbb{E}[x]}{T})$.
4. Generate T according to the cooling schedule and return to Step 1.

Figure 2.3. General Simulated Annealing Algorithm

$$P(E) = c_T \exp(-\frac{E}{c_b T}) \quad (2.8)$$

where $c_T > 0$ is the normalizing constant, $c_b > 0$ is the Boltzmann constant, and T is the temperature of the system.

The resulting search algorithm is presented in Figure 2.3, for a fixed temperature T , where system states are represented as points. The use of the Boltzmann-Gibbs distribution in the iterate selection subproblem provides a basis for the probabilistic acceptance of points which do not satisfy the simple decrease condition. After a large number of iterations, the system eventually reaches an equilibrium state governed by the Boltzmann-Gibbs energy distribution (Spall [54, p. 210]).

Formally introduced as an optimization method by Kirkpatrick *et al.* in [30], simulated annealing was developed as a method to find a good routing of wires on a circuit board. By varying the temperature T according to a user-defined cooling schedule, the Metropolis algorithm was extended to a general optimization technique. When the temperature is high, the algorithm is similar to a random walk and thus can globally explore the feasible

region. However, as the algorithm progresses, the temperature reduction restricts the acceptance iterates that do not strictly improve the quality of solution. Therefore, when the algorithm is applied to functions with many local optima, this modified Monte Carlo search technique has the ability to probabilistically find the global optimum.

Two key elements in the implementation of simulated annealing are the selection of candidate iterates and the cooling schedule. Selection of candidates iterate can be done randomly as shown in Figure 2.3; however, simulated annealing can perform more efficiently if the candidate iterate selection is based on an algorithm using available problem information. In such an implementation, the Boltzmann-Gibbs distribution would be used to allow the algorithm to make alternative selections for the best iterate. The selection of the cooling schedule represents a more complex issue. As discussed in the formation of crystals, if the cooling rate is too high, the system will not be able to fully explore the feasible region and may not find the global optimum. If the cooling rate is too low, the computational efficiency of the algorithm may be reduced. The selection of an appropriate cooling schedule and search algorithm are essential to the convergence of simulated annealing to a global optimum; however, in practice, these completely problem-dependent tuning parameters are not known. Therefore, simulated annealing requires problem information that is usually not available from a "black box" simulation to ensure convergence to a stationary point.

2.3.2 Evolutionary Algorithms

Evolutionary algorithms comprise a large set of numerical methods that uses adaptive (or evolutionary) processes to seek out optimal solutions. One main distinction of evolutionary algorithms is that they maintain a *population* of current "best" iterates rather than a single one, thus enabling them to explore many points in parallel instead of the common serial approach.

2.3.2.1 Genetic Algorithms. Perhaps the most popular of the evolutionary algorithms class are genetic algorithms (GA). Originally developed by Holland [26] to represent complex adaptive processes, ranging from biological systems to economies to political systems, genetic algorithms were based on principles of genetic variations and natural selection to mimic the evolution of a population over *generations*. The use of generations allows the whole population of genetic algorithms to be updated together; thus, the state of a GA is represented by all the individuals of the current population.

Using biological theories of competing traits, genetic algorithms can directly maintain control between trait dominance and randomness in the population genes (which from an optimization context is analogous to the search for local versus global optima). Because genetic algorithms are easily adapted to optimization problems and are able to rapidly locate good solutions, genetic algorithms have been accepted as an optimization tool.

Since genetic algorithms have a basis in natural selection, the terminology uses many biological names. For example, each iterate is referred to as an *individual* whose parameter values are called *chromosomes*. Although the chromosomes have traditionally been represented as binary strings, requiring standard bit (binary digit) coding, extensions of genetic algorithms to multiple character coding is an area of active research (Spall [54, p. 237]). The chromosomes of an individual are used to compute its *fitness* value, which is analogous to the objective function value of an iterate. The resulting fitness value can then be used to determine both an individual's inclusion in subsequent generations and matchings for offspring production (representing the "survival of the fittest" principle of natural selection). One distinction that GA makes from natural selection theory is the inclusion of an "elitism" strategy. De Jong, a doctoral student of Holland's who systematically studied the tuning of the genetic algorithm parameters, first described an "elitism"

strategy, which allows for asexual reproduction of individuals based on fitness values [18]. The "elite" individuals can be directly appended to the next generation and given priority during crossover matchings, thus ensuring the best chromosomes are preserved.

Genetic operators of *crossover* and *mutation* are then applied probabilistically to the current population to produce a new generation of individuals. During the crossover (or recombination) operation, the chromosomes of the individuals to be reproduced, referred to as the *parents*, are used in a recombination process to determine the chromosomes of the *offspring*. The usefulness of the recombination process depends on the nature of the fitness function. If the chromosomes operate independently of one another in the fitness evaluation, then crossover can quickly improve the offspring's fitness value. However, if the chromosomes are all intimately linked, the algorithm can skip the crossover operation and rely on mutation to seek fitness improvement. The mutation operation of genetic algorithms actively searches for improvements using the current population. In the basic formulation, the mutation operation randomly changes the chromosomes; however, just as with simulated annealing, genetic algorithms can perform more efficiently if the mutation is based on an algorithm using available problem information. The resulting general genetic algorithm is given in Figure 2.4.

Although there are several classes of evolutionary algorithms, genetic algorithms have been the most widely used. As an example of other evolutionary algorithms, evolution strategies are reviewed in the next subsection.

2.3.2.2 Evolution Strategies. Evolution strategies are specifically designed for constrained continuous variable optimization. Instead of extending genetic algorithms by changing the discrete chromosomal coding, evolutionary strategies were developed independently by Rechenberg [50] to use the natural (continuous) values of the individual's

General Genetic Algorithm

Initialization: Choose an initial population of N random individuals.

Evaluate the fitness function for each individual.

Set $N_e = 0$ if elitism strategy is used, otherwise select $0 < N_e < N$.

1. Select with replacement $N - N_e$ parents from the full population (including the N_e elite individuals). The selection of parents should employ a strategy to ensure individuals with higher fitness value are selected more often.
2. (Crossover) Apply a recombination process to the chromosomes of the parents to produce offspring.
3. While retaining the N_e elite individuals, replace the previous generation with the offspring generated in step 2.
4. (Mutation) Apply an appropriate algorithm to change the chromosomes of the population.
5. Compute the fitness values for the new population. Terminate the algorithm if the stopping criterion is met or if the budget of fitness function evaluations is reached; otherwise, return to step 1.

Figure 2.4. General Genetic Algorithm (adapted from Spall [54])

parameters. In Rechenberg's original work, the population was limited to a size of one but was extended with the introduction of crossover to populations with more than one individual. Other differences between genetic algorithms and evolution strategies are related to the basic genetic operations used to generate offspring and population sets. Before individuals are selected for inclusion in the current population, offspring are generated by mutating the parent chromosomes according to a zero mean probability distribution (usually a normal distribution) with a variance that is either user-defined or based on the covariance of the parent chromosomes. Offspring that do not meet the constraints of the problem are simply discarded. The resulting valid offspring compete either with or without the parents for inclusion in the next generation. The resulting evolutionary strategy algorithm is given in Figure 2.5.

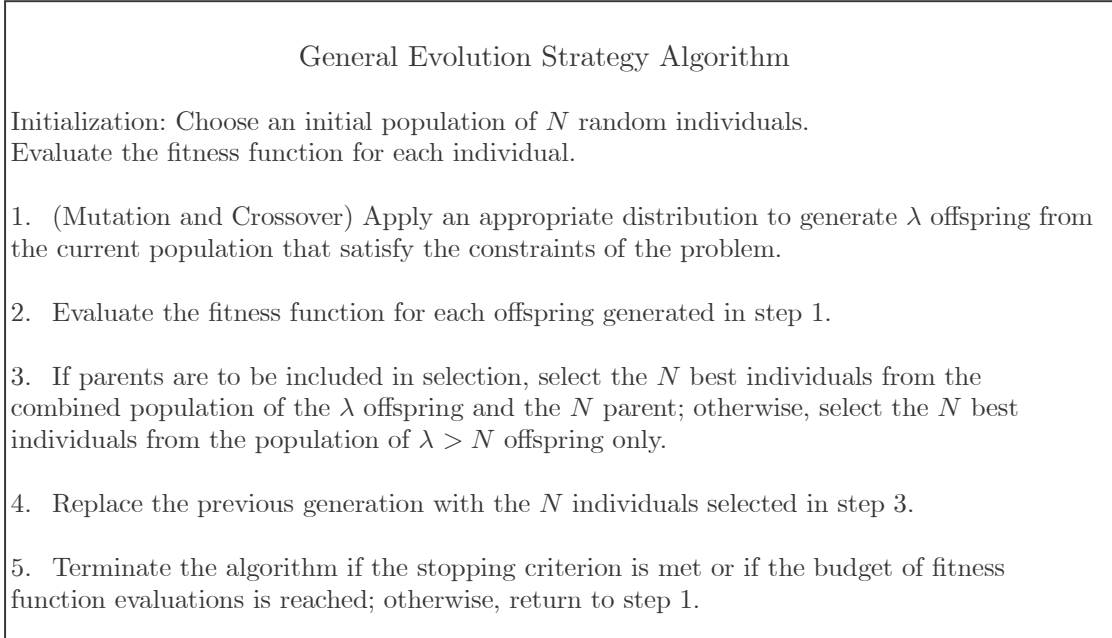


Figure 2.5. General Evolution Strategies Algorithm (adapted from Spall [54])

Just as with simulated annealing, the selection of appropriate algorithmic parameters and search algorithm is essential to the convergence of evolutionary algorithms to a global optimum. Although evolutionary algorithms, particularly genetic algorithms, have shown success in many diverse types of applications, they may perform significantly poorer than even random searches. The general lack of understanding in the performance of evolutionary algorithms has motivated a large portion of the current research to characterize what it is that makes certain problems hard for these algorithms.

2.3.3 Suitability of Search Heuristics

Through use of embedded stochastic processes, many of the search heuristics can avoid local optima. The balance between generality and performance properties for search heuristic methods provides a good example of the no free lunch (NFL) theorem. According to the NFL theorem by Wolpert and Macready [63], if an algorithm does particularly well on average over one class of problems then it must do worse on average over the

remaining problems. Thus, when a comparison of performance is conducted on a class of problems between a general search heuristic and a tailored algorithm, the tailored algorithm frequently outperforms the search heuristic. In the case of search heuristics, their ability to perform generally well over a diverse set of problems infers that they may not perform well on individual problem classes.

When considering the development of an algorithm to solve simulation optimization problems, the underlying algorithm needs to be general enough to be applicable to a variety of simulations yet narrow enough in scope to avoid issues the NFL theorem presents. Additionally, the underlying algorithm's convergence theory should be robust enough to avoid overtailoring the algorithm, since simulations are often "black box" systems. Because the convergence theory and algorithmic performance are generally unreliable, search heuristics are not used in this research.

2.4 Generalized Pattern Search (GPS) Methods

As shown in Section 2.1, the use of the general optimization algorithm in Figure 2.1 does not ensure convergence to a stationary point of the optimization problem given in Equation (2.1). Pattern search methods, a subclass of direct search methods, have been proven convergent to a stationary point by the use of a conceptual mesh. Additionally, pattern search methods have been modified for applicability to MVP problems. In this section, the mesh conditions placed on the pattern search methods and adaptations of GPS methods are discussed in relation to the underlying convergence theory.

2.4.1 Pattern Search Methods

Before 1997, the direct search methods were generally viewed as search heuristics because they lacked convergence theory. Torczon [58] made a significant contribution by not only demonstrating that many of the seemingly diverse direct search methods members

could be unified under a generalized subclass, designated as pattern search methods, but under reasonable assumptions, this new subclass of methods was proven convergent to a stationary point. Specifically, if the objective function f is continuously differentiable in a neighborhood of the *level set*,

$$L(x_0) = \{x : f(x) \leq f(x_0)\} \quad (2.9)$$

then a subsequence of GPS iterates $\{x_k\}$ will globally converge to a point \hat{x} , satisfying $\nabla f(\hat{x}) = 0$; that is, $\liminf_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0$. Thus, without using any derivative information, pattern search is globally convergent to a stationary point. In order to unify the members of the pattern search subclass, Torczon [58] imposed requirements on both the location of candidate iterates and the update operation used in the direct search methods. The candidate iterates eligible to replace the current incumbent are required to lie on a conceptual lattice (or *mesh*), which is determined by the direction set and a step length parameter maintained by the algorithm.

Once the set of candidate points is established, each point is evaluated by the objective function and compared against the incumbent. If improvement is found, then the successful candidate iterate is accepted as the new incumbent and step size is either maintained or increased (which retains or coarsens the mesh, respectively); otherwise, the step size is decreased (which refines the mesh) and the members of the new candidate iterate set are evaluated for success. It is important to note that the use of the mesh makes satisfaction of a sufficient decrease condition, Equation (2.3), automatic, thus only simple decrease is required (Kolda *et al.* [31]). By inspection, the conditions of convergence for the pattern search methods are analogous to the conditions required in Section 2.1. In particular, the use of a positive spanning set (described in Section 3.1) satisfies the angle condition

and the updating operation of the step size ensures that the steps are neither too long nor too short, which was previously ensured by the use of sufficient decrease and either the curvature condition or backtracking.

Lewis and Torczon [34] further generalized the algorithm applying the theory of positive linear dependence presented by Davis [17]. The direction set is selected in the generalized algorithm so that it forms a positive spanning set for the domain of the optimization function. A set of vectors $\{a_1, \dots, a_p\}$ positively spans \mathbb{R}^n if any vector $x \in \mathbb{R}^n$ can be written as a nonnegative linear combination of the vectors in the set; *i.e.*,

$$x = \lambda_1 a_1 + \dots + \lambda_p a_p \tag{2.10}$$

where $\lambda_i \geq 0$ for $\forall i = 1, \dots, p$. Through this construction of the candidate iterate set, whenever $\nabla f(x) \neq 0$, where x is the current incumbent, a descent direction of the objective function can be captured in at least one member of the candidate iterate set. The set $\{a_1, \dots, a_p\}$ is called positively dependent if one of the a_i 's is a nonnegative combination of the others; otherwise, the set is positively independent. Since a positive basis is a positively independent set whose positive span is \mathbb{R}^n , a positive basis is the smallest proper subset of a positive spanning set that still positively spans \mathbb{R}^n . Davis showed that any positive basis in \mathbb{R}^n contains between $n + 1$ and $2n$ elements (referred to as a minimal and maximal set, respectively); therefore, Lewis and Torczon bound the worst case number of members in the candidate set per iteration to $n + 1$ iterates. As noted in Dolan *et al.* [21], "the key to the global analysis is the notion of having searched in a sufficient number of directions from the current iterate to guarantee that we have not overlooked a potential direction of descent." Thus, the use of a positive basis enables the algorithm to use the fewest number of function evaluations without the accidental avoidance of descent directions.

Pattern search methods unified many of the most popular direct search methods, including the coordinate search with fixed step lengths, the original Hooke-Jeeves method [28], EVOP with factorial designs (Box [11]), and multidirectional search (Dennis and Torczon [20]). However, it should be noted that one of the first and most popular methods, the original Nelder-Mead (downhill simplex) method [44], does not fall into the class of pattern search methods. The original Nelder-Mead method is based on the use of a simplex, which is a geometrical figure that has one more vertex than dimension, to capture first-order information. The method uses a line search along the line passing through the centroid and the vertex that has the currently least favorable objective value. The line search attempts to locate a new point for the vertex that produces an objective function value that is strictly better than that of the second least favorable vertex. If such a point is found, the vertex is moved to the new point (thus, deforming the shape of the simplex) and the method repeats with the new least favorable vertex. If the line search is unsuccessful in locating a new point, all the vertices are contracted toward the vertex with the most favorable objective function value. The result of this relatively simplistic method is a simplex that is able to quickly adapt to the local topology of the function. However, as detailed by Kolda *et al.* [31], the unmodified version of this method cannot be considered a GPS method because the selection of the candidate set does not ensure an existing descent direction is located and the update operation is based on comparisons to the second least favorable vertex instead of the current incumbent value.

2.4.2 Extensions to Constrained Problems

Lewis and Torczon ([35] and [36]) showed that placing an additional requirement on the candidate iterate set enables the convergence theory established in the unconstrained GPS algorithm to be extended to bound and linearly constrained optimization with continuously

differentiable objective functions. Specifically, a subsequence of GPS iterates converges to a point \hat{x} satisfying the first-order necessary condition for optimality; namely, $\nabla f(\hat{x})^T(x - \hat{x}) \geq 0$ for all feasible x . The additional requirement is that the set of directions that defines the candidate iterate set must be chosen to conform to the geometric boundary of the active constraints. In an effort to keep all the iterates of the GPS methods feasible (thereby producing *feasible point* methods) Lewis and Torczon required that the initial iterate be a feasible point and used an *exact penalization approach*; namely,

$$\min F(x) \tag{2.11}$$

$$\text{where } F(x) = \begin{cases} f(x), & \text{if } x \in \Omega \\ +\infty, & \text{otherwise} \end{cases}$$

and $\Omega = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$ or $\Omega = \{x \in \mathbb{R}^n \mid l \leq Ax \leq u\}$ for bound and linearly constrained optimization problems, respectively.

Audet and Dennis [6] presented the unification of the unconstrained, bound, and linearly constrained versions of GPS methods by applying a barrier function directly to the function instead of Lewis and Torczon's approach of making the use of an exact penalization approach dependent on the selection of GPS algorithm. The *barrier approach* simply replaces the optimization function of the original problem with $f_\Omega(x)$, which is given by

$$f_\Omega(x) = \begin{cases} f(x), & \text{if } x \in \Omega \\ +\infty, & \text{otherwise} \end{cases} \tag{2.12}$$

where Ω is the feasible region. By considering all the previous GPS algorithms as the same algorithm with the barrier function used as the optimization function, the analysis and theoretical results are applicable to all the previous methods. Similar to the previous work, the step size (*mesh size*) parameter is updated conditionally on the replacement of

the incumbent and the candidate iterates are required to lie on the mesh M_k , which is given by

$$M_k = \{x_k + \Delta_k D z : z \in \mathbb{Z}_+^{|D|}\} \quad (2.13)$$

where $x_k \in \mathbb{R}^{n^c}$ is the current iterate, $D \in \mathbb{R}^{n^c \times |D|}$ is a positive spanning set, and $\Delta_k \in \mathbb{R}$ is the mesh size parameter. Only the following additional rule needs to be enforced to assure convergence to a stationary point: each direction $d_j \in D$, for $j = 1, \dots, |D|$, must be of the form $d = G\bar{z}$, where $\bar{z} \in \mathbb{Z}^{n^c}$ and $G \in \mathbb{Z}^{n^c \times n^c}$ is a nonsingular *generating* matrix.

In order to allow the user the flexibility to incorporate search heuristics (see Booker *et al.* [9] and [10]), Audet and Dennis ([6]) explicitly separate out a SEARCH step to complement the POLL step, which is different but equivalent to the work of Torczon ([58]). During the POLL step, the members of the poll set (candidate set) are evaluated and compared to the incumbent for improvement. The poll set P_k is composed of mesh points neighboring the current incumbent x_k in the directions of the columns of $D_k \subseteq D$; thus the poll set can be expressed as

$$P_k = \{x_k + \Delta_k d : d \in D_k\} \quad (2.14)$$

where D_k is the current positive spanning set. Thus, there is great freedom in choosing the directions of the positive spanning set.

Since the convergence results do not depend on the SEARCH step, the user is provided this optional step to employ a finite heuristic to accelerate the convergence of the algorithm. If either the SEARCH or POLL step produces an *improved mesh point*, then the current iteration can end, the mesh size parameter is kept the same or is increased, the improved mesh point becomes the new incumbent, and the process is reiterated. However, if both the

General Pattern Search Algorithm

Initialization: Choose a feasible starting point x_0 such that $f_\Omega(x_0) < \infty$.

Let D be a positive spanning set.

Let the $M_0 \subset \Omega$ be the mesh defined by mesh size parameter $\Delta_0 > 0$ and $D_0 \in D$

Set the iteration counter k to 0.

For $k = 0, 1, \dots$

1. SEARCH Step (Optional): Employ some finite strategy seeking an improved mesh point; *i.e.*, $x_{k+1} \in M_k$ such that $f_\Omega(x_{k+1}) < f_\Omega(x_k)$.

2. POLL Step: If the SEARCH step does not find an improved mesh point, evaluate f_Ω at the points in POLL set P_k until an improved mesh point x_{k+1} is found (or until done).

3. Update: If SEARCH or POLL finds an improved mesh point, then update x_{k+1} , and set $\Delta_{k+1} = \tau^{m_k} \Delta_k \geq \Delta_k$ where $\tau > 1$ is a rational number that remains constant over all iterations, and the integer m_k satisfies $0 \leq m_k \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$;

Otherwise, set $x_{k+1} = x_k$, and set $\Delta_{k+1} = \tau^{m_k} \Delta_k < \Delta_k$ where τ is a rational number that remains constant over all iterations, $\tau^{m_k} \in (0, 1)$, and the integer m_k satisfies $m_{\min} \leq m_k \leq -1$ for some fixed integer m_{\min} .

4. Terminate the algorithm if the stopping criterion is met or if the budget of function evaluations is reached; otherwise, return to step 1.

Figure 2.6. General Pattern Search Algorithm (adapted from Audet and Dennis [6])

SEARCH or POLL step fail to produce an improvement, then the incumbent is declared to be a *mesh local optimizer*, the mesh size parameter is decreased, and the process is reiterated.

Although not directly covered in this research, pattern search methods have also been extended to problems with nonlinear constraints. Lewis and Torczon [37] apply GPS for bound constraints to an augmented Lagrangian (see Conn *et al.* [13]) formulated from the constraints of the problem. By iteratively reducing the terminating mesh size parameter and using the results of one iteration to provide parameter estimates for the next, the algorithm converges to a stationary point under the assumption of twice continuous differentiability of the objective and constraint functions. The drawback in practice is that performance is driven by a problem-dependent penalty parameter.

Audet and Dennis [8] alternatively extend GPS for nonlinear constraints by incorporating a filter. The filter algorithm was first introduced by Fletcher and Leyffer [23] as a method to globalize sequential quadratic programming (SQP) and sequential linear programming (SLP) without requiring the user to specify the weight parameters used in an alternative merit function approach. A filter algorithm introduces a constraint violation function that aggregates constraint violations for sampled infeasible points and uses a bi-objective approach to the optimization problem, in which the filter accepts an iterate that improves either the objective function or the aggregate constraint violation function. While convergence to a point satisfying first order optimality conditions is not proven by the use of the resulting algorithm (since the convergence results that are guaranteed depend strongly on the set of directions used in the POLL step), it is demonstrated that as a richer set of directions are used, the likelihood of achieving a point satisfying first-order optimality conditions increases. The pattern search filter algorithm reduces to the basic GPS algorithm when nonlinear constraints are absent. It does not use derivative information, requires only simple decrease in function value for a mesh parameter update, and does not require any constraint qualifications on the nonlinear constraints.

In [7], Audet and Dennis introduce the Mesh Adaptive Direct Search (MADS) algorithm to handle general constraints. MADS has a similar structure to GPS; however, MADS introduces a poll size parameter Δ_k^p that is used in conjunction with the mesh size parameter Δ_k^m , the current incumbent x_k , and a positive spanning matrix D_k to construct a *frame* (the former POLL set, renamed to align with the Coope and Price nomenclature in [14]) used during the POLL step. Although both the poll size and mesh size parameters are used to construct the frame, it is important to note the poll size parameter does not directly influence the mesh. This distinction allows the algorithm to choose from a richer pool of

candidate sets for the direction by keeping $\Delta_k^p \geq \Delta_k^m$. Thus, by letting Δ_k^m go to zero more rapidly than Δ_k^p , the directions in D_k used to define the frame may then be selected from a larger set of directions that become dense in the limit. When combined with the barrier method of assigning a function value of infinity to infeasible iterates, MADS has stronger convergence properties than the original GPS algorithm, even with nonlinear constraints.

2.4.3 General Pattern Search Extensions

Material to this research was the development of a framework for MVP problems with linear constraints by Audet and Dennis [5]. Having already defined a local POLL in the GPS algorithm, the Mixed Variable Generalized Pattern Search (MGPS) accommodates categorical variables by dividing polling into three stages. The local POLL of the GPS algorithm for continuous variables is augmented with polling of the current set of discrete neighbors and extended polling to explore around promising neighbors, which is discussed in Chapter 3. Abramson [2] extended MGPS to general nonlinear constraints by using a filter. This is pertinent to this research because it is incorporated into the NOMADm MATLAB[®] software [1].

It is also important to mention the significant contributions made in the application-related research for GPS algorithms. Although the proximity of linear constraints requiring the direction set D_k to conform to the geometry of the boundary is presented by Lewis and Torczon [36], the algorithm presented for constructing the required directions did not directly address the case of linearly dependent active constraints. Brezhneva and Dennis [12] provide a general algorithm that first identifies nonredundant active constraints via a projection approach and the solution of a linear program and then constructs sets D_k taking into account only nonredundant constraints.

To further characterize the quality of solutions that can be attained by GPS methods, Abramson [3] is able to prove under mild conditions some limited second-order behavior results. Even without first-order derivatives, under mild assumptions, Abramson proves that GPS cannot converge to a strict local maximizer, and with the additional assumption that the objective function f is sufficiently smooth, that the use of a $2n$ orthonormal positive basis prevents convergence to a saddle point where the sum of the eigenvalues of the Hessian are negative.

Since computational efficiency can be a driving factor in the selection of an optimization algorithm (particularly for direct search methods), methods for increasing efficiency by using available information are usually an actively investigated research topic for any computational field. Abramson *et al.* [4] demonstrate that the use of any derivative information can significantly reduce the overall number of function evaluations required of a GPS algorithm while maintaining the known convergence properties. The method uses any available gradient information to prune or remove directions of known ascent from the positive spanning set D_k , thus producing a pruned set D_k^P . The pruned set D_k^P is then substituted for the original positive spanning set in the standard POLL step of the GPS algorithm. As a result, even rather rough approximations to the gradient (on the mesh) are shown sufficient to reduce the poll set to a singleton, thus requiring only a single function evaluation at each POLL step.

Computational efficiency has also been shown to be improved by using gradient information calculated from a simplex of previously stored iterate responses (Custódio and Vicente [16]). In its most basic form, the simplex gradient can be used to control poll ordering in an effort to reduce the number of function evaluations within a POLL step before an improved mesh point is located. If the simplex gradient is used to approximate

the true gradient, pruning can again lead to a singleton in the poll set. Furthermore, the simplex Hessian can be constructed and used to generate an approximation to the Newton direction. Simplex derivative information can also be used to form a candidate set to be evaluated during the SEARCH step or to impose a sufficient decrease condition on the update of the mesh size parameter. Thus, without requiring additional function evaluations, the use of simplex derivatives can be used successfully to increase the computational efficiency of the GPS algorithm.

Kolda *et al.* [31] introduced the class of generating search set (GSS) methods that contains not only pattern search, but also moving grids (algorithms inspired by Coope and Price [14] that conditionally enable the mesh to change between iterations), and methods that directly enforce the sufficient decrease condition of Equation (2.5) without derivative information, such as those found in [39] by Lucidi and Sciandrone, instead of using the simple decrease of Equation (2.3) with mesh size controls. In order to prove convergence, the GSS methods are unified under three principles (which are the same requirements of the derivative-based methods discussed in Section 2.2):

1. the algorithm must have a search direction that is a descent direction, *i.e.* a search direction p_k where $p_k^T \nabla f(x_k) < 0$
2. GSS methods must avoid poor search directions, *i.e.* search directions must satisfy the sufficient descent condition of Equation (2.4)
3. GSS methods must avoid poor choices of step lengths; *i.e.* the steps taken must not be too long or too short

Principles 1 and 2 are necessarily satisfied for all members of the GSS class because the use of generating sets that positively span the domain of the objective function ensure that at least one direction is a descent direction that is not orthogonal to the gradient. Since each of the GSS methods decreases the mesh size parameter after unsuccessful iterations, this is equivalent to a multidirectional line search with backtracking which will ensure the

steps are not too short. As a result, the only requirement for a GSS method to converge to a stationary point is to have a means to avoid choices of step lengths that are too long. In order to allow for the inclusion of various algorithms within the GSS class, the condition for appropriate step lengths is generalized as

$$f(x_k + \Delta_k p_k) \leq f(x_k) - \rho(\Delta_k) \quad (2.15)$$

where the *forcing function* ρ is used to change the inequality between the simple decrease condition and the sufficient decrease condition. For algorithms that use the mesh update to limit the maximum length of the step size, such as GPS and moving grids, only the simple decrease condition needs to be fulfilled. For these algorithms, the forcing function is simply set to zero to produce the simple decrease condition of Equation (2.3). For algorithms that do not restrict the maximum step size length, the sufficient decrease condition is created by requiring ρ to be a continuous, monotonically increasing function that is linear as Δ_k decreases to zero. Kolda *et al.* [31] demonstrate that every member of the GSS class is able to satisfy the conditions, thereby proving convergence for GSS methods.

2.4.4 Suitability of Generalized Pattern Search (GPS) Methods

Generalized Pattern Search methods are well-suited for use in the development of an algorithm to solve simulation optimization problems. Since they are a subclass of direct search methods that do not use derivative information, GPS methods are directly applicable to problems where derivative information is unavailable and cannot be accurately calculated. Additionally, GPS methods have been proved convergent not only as a class itself, but also as a member of the superclass of GSS methods. The flexibility of GPS methods has been demonstrated by tailoring to allow for the handling of different types of constraints and variables (to include MVP problems).

Although all members of the GSS class are proven to be convergent to a stationary point, the generality of the problem structure and level of development are important factors in the selection of the underlying optimization algorithm. For example, the moving grid algorithm presented by Coope and Price allows greater flexibility of the mesh and could be extended to handle categorical variables, but their algorithm has not been encoded for general use. Since NOMADm is an algorithm that can handle MVP problems, the underlying MGPS algorithm of NOMADm will be modified in this research to develop an algorithm that can efficiently solve simulation-based optimization problems of "black box" systems.

2.5 Implementation Considerations

With the underlying algorithm selected, the adaptation of the current algorithm to stochastic simulations and computational efficiency improvements may be discussed. In this section, the use of ranking and selection procedures and surrogate searches will be discussed as they relate to efficiently solving simulation-based optimization problems. It will be shown that the key concept in both modifications is to gather as much additional problem information as possible without requiring a large number of additional function evaluations.

By considering the stochastic simulation as a response generator, a typical approach in reducing the sampling error of the expected performance is to replicate the design vector a sufficiently large number of times. This approach is justified by the weak law of large numbers, since the sample mean will equal the population mean as the number of samples goes to infinity. In the context of a simulation optimization algorithm, the total number of function evaluations is traditionally a limiting factor and thus a trade-off must usually be made between improving the solution and controlling the sampling error. In this research,

this balance will be maintained through the use of statistical procedures that require a minimal number of function evaluations for a given confidence level, thereby preserving function evaluations for iterate improvement.

Since the selected algorithm relies on comparisons between stochastic responses, a review of statistical procedures for detecting these differences is warranted. Consider a set of k candidate design vectors $\{X_1, \dots, X_k\}$ under consideration with true objective function values denoted $f_i = f(X_i) = \mathbb{E}[F(X_i, \psi)]$, where $i = 1, \dots, k$. The candidate with the i th best true objective function value will be denoted by $X_{[i]}$, with associated true objective function $f_{[i]}$. The desired procedure is one that guarantees selection of $X_{[1]}$ with a user-specified probability of at least $1 - \alpha$, which can be expressed as

$$\Pr\{\text{select } X_{[1]} | f_{[i]} - f_{[1]} \geq 0, i = 2, \dots, k\} \geq 1 - \alpha \quad (2.16)$$

In the case where there are only two candidate design vectors, one can simply use an appropriate *pairwise comparison* test (t -test) to perform the selection at the given level of significance. However, when there are at least three candidates, *multiple comparison* tests may be required.

When attempting to determine which candidate provides the best true objective function value, a common first step in multiple comparison testing is to first determine if there is a statistically significant difference between the candidates. Assuming that the noise in each of the responses is independently and identically distributed (i.i.d) normally with mean zero, the Tukey-Kramer method is a generalization of the pairwise t -test that produces a common acceptance interval for the differences between sample mean responses for each member pair of the candidate set. Since the null hypothesis of the test is that all the candidates have the same mean, there is enough statistical evidence at the given α level

of significance to state that at least one candidate has a different mean value if any of the $k(k-1)/2$ pairs of mean differences lie outside of the acceptance interval. Although this result is rather broad, it may provide an indication of which candidate(s) provide a better response.

After a test of equal means has been performed, multiple comparison testing attempts to find the candidate that provides the best true objective function value by eliminating inferior candidates. A standard method to perform this culling of candidates is through multiple comparison with the best (MCB) testing. By assuming some level of prior information about which candidate is the optimum, there are only $k-1$ pairs of candidates that require one-way testing, as opposed to the two-way testing of $k(k-1)/2$ pairs of candidates in the Tukey-Kramer method. Therefore, MCB tests can provide stronger results than those of the Tukey-Kramer method.

The MCB test can be constructed with a null hypothesis in which all the candidates produce responses at least as good as the assumed optimum. Thus, just as with the first multiple comparison test, the desire is to reject this hypothesis. For the assumed optimum to be the true best, the difference in sample mean responses between the optimum and each candidate must lie within an "appropriate" acceptance region, which is calculated from available information. Thus, if E_i represents the event that the difference in sample mean responses of the optimum and the candidate i lies within the acceptance region (that candidate i produces a response at least as favorable as the optimum), then the event E that the optimum is in the overall acceptance region (that all the candidates produce a response at least as favorable as the optimum) is given by $E_1 \cap \dots \cap E_k$. Thus, the probability that the optimum is the best is given by

$$\Pr\{E\} = \Pr\{E_1 \cap \dots \cap E_k\} \quad (2.17)$$

Since Equation (2.16) requires that $\Pr\{E\} \geq 1 - \alpha$, the joint probability is given by

$$\Pr\{E_1 \cap \dots \cap E_k\} \geq 1 - \alpha \quad (2.18)$$

The acceptance regions for each of the individual candidates can be calculated by partitioning the level of confidence α among the candidates. The most straightforward manner is through the Bonferroni inequality, which produces an acceptance region bound of

$$\Pr\{E_i\} = (1 - \alpha)/(k - 1) \quad (2.19)$$

for each candidate. If the additional assumption that each of the differences in sample mean responses between the optimum and the candidate are jointly normally distributed with mean zero is maintained, then the acceptance region bound can be improved through the use of the Slepian inequality (see Tong [57, p.8]). The resulting acceptance region bound for each candidate is given by

$$\Pr\{E_i\} = (1 - \alpha)^{1/(k-1)} \quad (2.20)$$

Once the individual candidate acceptance regions are formed from the derived level of confidence and the sample variances, the difference in each sample mean response between the optimum and the candidate is tested against its acceptance bound. If the null hypothesis is rejected and each of the individual acceptance bounds are broken, then strong evidence exists that the assumed optimum is the true best.

From this review, it is apparent that the type of test used for guaranteed correct selection of the best candidate is dependent on the amount of information available and simplifying assumptions taken. With this foundation, the development of ranking and selection procedures and how they apply to this research may be discussed. Trosset [60] demonstrated that although optimization in the presence of random noise is more difficult than a noise-free case, statistical testing for iterate selection can be used in pattern search methods to maintain convergence results. However to do so, the number of replications per iteration increases faster than the squared reciprocal of the mesh size parameter as the sequence converges (Trosset [60]). Sriver [55] established an algorithmic framework for GPS methods with stochastic responses and proposed ranking and selection (R&S) methods for the statistical testing of GPS iterates. Sriver [55] overcomes the problems identified by Trosset [60] through the use of an indifference zone.

Ranking and selection methods are slightly different from multiple comparison tests. The goal of MCBs is to characterize the members of the candidate set; whereas, the goal of R&S methods is to actively screen the k candidates to find a subset likely to contain the optimum. MCBs are generally a tool of inference, where the confidence intervals provide information on how close the systems may be to one another. The resulting confidence intervals can be combined with other factors in selecting an overall best candidate. R&S methods are a tool for finding the optimum set without the need to characterize the candidate set. Since the focus of this research is strictly concerned with finding improvement, R&S methods are more applicable.

Proposed by Paulson [47], fully sequential R&S procedures with indifference zone use a *triangular continuation region* to determine when enough information has been gathered to find the optimum. Indifference zone procedures differ slightly with the probability

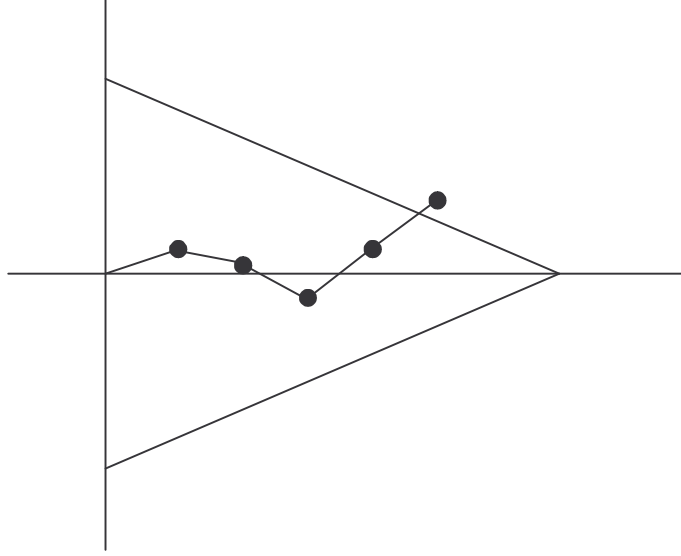


Figure 2.7. Triangular Continuation Region

statement in Equation (2.16) by the use of an *indifference parameter* $\delta > 0$. This user-selected parameter is essentially the smallest "important" difference between solutions, often referred to as the practical difference. That is, points with function values within δ of each other are considered equivalent. These procedures select $X_{[i]}$ with a user-specified probability of at least $1 - \alpha$ whenever the difference is worth detecting, which can be expressed as

$$\Pr\{\text{select } X_{[i]} | f_{[i]} - f_{[1]} \leq \delta, i = 2, \dots, k\} \geq 1 - \alpha \quad (2.21)$$

By approximating the sum of differences between two systems as Brownian motion, Paulson developed a triangular continuation region, illustrated in Figure 2.7, that acts in part as an acceptance region of the above comparative tests. The procedure iteratively sums the difference between the two systems as long as the result lies within the triangular region. The sum can leave the triangular region in three ways. If the sum crosses the acceptance bound, then the system associated with the null hypothesis is declared the optimum. If the

sum crosses the rejection bound, then the system associated with the alternative hypothesis is declared the optimum. Otherwise, if the sum exits without triggering a bound, the system with the more favorable average responses is declared the optimum. Thus, the triangular region indicates when one system is clearly superior to the other, resulting in a selection of a candidate that satisfies Equation (2.21).

The original R&S procedure has undergone modification to increase efficiency by using tighter bounds, similar to the use of the Slepian inequality for MCBs acceptance bound. Hartmann [25] improved upon Paulson’s procedure by replacing Boole’s inequality with a geometric inequality and using a Brownian motion bound on the acceptance region instead of a large deviation bound under the assumption of normal and common variance noise. Kim and Nelson [29] further extend Hartmann’s work by the handling of unequal and unknown variances. Pichitlamken and Nelson [48] introduced Sequential Selection with Memory (SSM), which is similar to the Kim and Nelson procedure but includes previously sampled responses in the current R&S procedure. Srivier [55] demonstrated that SSM performed well with GPS methods in controlling the number of required function evaluations while guaranteeing the proper level of confidence in the iterate selection. Through the use of an efficient R&S procedure, the MGPS algorithm within NOMADm can be extended to stochastic simulation problems without requiring a prohibitive number of function evaluations; therefore, SSM is used in this research to control the selection of iterates.

In addition to modifying the current NOMADm code to handle stochastic simulations, this research seeks to meet the goals of increasing the computational efficiency of the optimization algorithm and performance testing of appropriate surrogates. As noted by Booker *et al.* [10], surrogates that serve as approximations of an expensive simulation, constructed by interpolating or smoothing known values of the objective, can be used to reduce the

number of function evaluations required to discover a point that produces decrease. Additionally, if the response surface has more than one optimum, the use of surrogates could lead to stationary points with better objective function values. Therefore, this research examines the use of surrogates for both local and global searches, which is discussed in more detail in Chapter 4. In particular, Latin hypercube sampling (LHS) will be used as an initial space-filling global search while the local search will be aided either by the use of a parametric model-fitting approach of Kriging or a nonparametric fitting method of kernel regression.

2.6 Summary

After reviewing the requirements for optimization algorithms to prove convergence to a stationary point, the GPS class of algorithms are shown to adhere to the same requirements for convergence as the gradient-based methods. However, since the GPS methods are suitable for simulation optimization (only requiring simple decrease in objective function evaluation by the use of a mesh and having been adapted to mixed variable domains), a GPS method is used in this research. The next chapter presents the approach for using GPS with ranking and selection procedures.

Chapter 3 - Approach

This chapter discusses in more detail the fundamental concepts of the MGPS algorithm and the ranking and selection procedure used in this research. First, the construction of the mesh and the poll set of pattern search methods are adapted for mixed variable domains in a way that maintains convergence properties. Then, a method for handling bound and linear constraints within MGPS algorithm is described. Finally, the proposed algorithm modifications are presented with associated implementation considerations.

3.1 Mesh and Poll Set Construction

As noted in Chapter 2, the MGPS algorithm is related to was developed from the GPS algorithm of Audet and Dennis [6]; thus there are common restrictions imposed on the iterates to ensure proven convergence to a stationary point. In particular, at any iteration k , all iterates are required to lie on the mesh M_k . Although the mesh is not actually constructed, the concept of the mesh is essential to the construction of the POLL set from which candidate iterates are generated. In order to define the mesh in the context of MVP problems, first recall the notation presented in Equation (1.3): each vector $x \in \Omega$ can be denoted as $x = (x^c, x^d)$ where $x^c \in \mathbb{R}^{n^c}$ are the continuous variables of dimension n^c and $x^d \in \mathbb{Z}^{n^d}$ are the discrete variables of dimension n^d . By fixing the values of the integer space \mathbb{Z}^{n^d} , a *discrete plane* can be defined as a section of the mesh where only the continuous values may vary. Additionally, since the predefined categorical list is finite, each discrete plane i , $i = 1, \dots, i_{\max}$, can be identified uniquely.

The mesh construction is formulated to provide enough generality to be applicable to MVP problems, yet revert back to the single discrete plane mesh structure for GPS methods given in Equation (2.13). In order to guarantee convergence in GPS methods, the set of directions used to generate the POLL set must be sufficiently rich to produce a

descent direction. This condition is met by requiring the direction set to positively span the domain of the objective function f . The following definitions introduced by Davis [17] provide the foundation for the use of positive spanning sets within GPS methods:

Definition 3.1 A *positive combination* of the set of vectors $V = \{v_i\}_{i=1}^r$ is a linear combination $\sum_{i=1}^r c_i v_i$, where $c_i \geq 0$, $i = 1, 2, \dots, r$.

Definition 3.2 A finite set of vectors $W = \{w_i\}_{i=1}^r$ forms a *positive spanning set* for \mathbb{R}^n if every $v \in \mathbb{R}^n$ can be expressed as a positive combination of vectors in W . The set of vectors W is said to *positively span* \mathbb{R}^n .

Definition 3.3 A positive spanning set of vectors W is said to be a *positive basis* for \mathbb{R}^n if no proper subset of W positively spans \mathbb{R}^n .

Davis [17] proved that the cardinality of any positive basis in \mathbb{R}^n is between $n + 1$ (a minimal set) and $2n$ (a maximal set). Examples of minimal and maximal sets in matrix notation can be quickly generated respectively by using the associated matrices $[I; -e]$ and $[I; -I]$, where I is the identity matrix and e is the vector of ones, and are commonly selected as the positive bases. The key purpose in using positive spanning sets in GPS is derived from the theorem:

Theorem 3.4 (Davis [17]) A set D positively spans \mathbb{R}^n if and only if, for all nonzero $v \in \mathbb{R}^n$, $v^T d > 0$ for some $d \in D$.

Thus, if a positive spanning set D is constructed at a point x where the gradient $\nabla f(x)$ exists and is nonzero, then by Theorem 3.4 at least one element of D is required to be a descent direction (since $\nabla f(x)^T d < 0$ for some $d \in D$ and $v = -\nabla f(x)$).

For each discrete plane $i = 1, \dots, i_{\max}$, a set of positive spanning directions $D^i \in \mathbb{R}^{n^c \times |D^i|}$ is constructed by forming the product

$$D^i = G_i Z_i \quad (3.1)$$

where $Z_i \in \mathbb{R}^{n^c \times |D^i|}$ and $G_i \in \mathbb{R}^{n^c \times n^c}$ is a nonsingular matrix for the discrete plane. (The notation is abused here and throughout this document, in that the set D^i is represented in Equation (3.1) as a matrix whose columns are elements of the set.) This additional restriction is required for convergence and is discussed in more detail by Torczon [58]. Although a common choice for G_i is the identity matrix, Audet and Dennis [5] note that the generating matrices G_i (and thus D^i) that define the discrete plane meshes can be determined as the iteration unfolds as long as only a finite number of them are generated. Therefore, the use of Equation (3.1) in restricting the selection of positive spanning directions maintains the original flexibility from the GPS methods, but allows for MVP problem structures and reduces to the single discrete plane version used in GPS methods when discrete variables are absent.

The mesh is formed as the direct product of X^d with the union of the local meshes for each possible categorical combination setting, which can be expressed as

$$M_k = X^d \times \bigcup_{i=1}^{i_{\max}} \{x_k^c + \Delta_k D^i z \in X^c : z \in \mathbb{Z}_+^{|D^i|}\} \quad (3.2)$$

where $x_k^c \in \mathbb{R}^{n^c}$ is the current iterate and Δ_k is the mesh size parameter. The POLL set P_k is composed of the continuous neighbors of the current incumbent x_k in the directions of the columns of D_k^i ; thus the POLL set can be expressed as

$$P_k = \{x_k + \Delta_k(d, 0) \in \Omega : d \in D_k^i\} \quad (3.3)$$

where the current iterate $x_k \in \Omega$ lies in the discrete plane i , $\Delta_k \in \mathbb{R}$ is the mesh size parameter, and $D_k^i \subseteq D^i$ is the current positive spanning set of the discrete plane i . The notation $(d, 0)$ indicates that only the continuous variables may change; thus,

$$x_k + \Delta_k(d, 0) = (x_k^c + \Delta_k d, x_k^d).$$

3.2 Optimality for Mixed Variable Domains

In order to properly apply optimization algorithms for MVP problems, some basic conceptual definitions used in optimization must be adapted to mixed variable domains. For example, the notion of local optimality must be revised to take into account the discrete planes illustrated by the structure of the mesh. Because continuous and (traditional) discrete variables can be represented as ordered sets, local optimality is well-defined in terms of local neighborhoods; however, for categorical variables, the concept of a local neighborhood must be defined by the user. Abramson [2] defines the set of discrete neighbors in terms of a set-valued function as $\mathcal{N} : \Omega \longrightarrow 2^\Omega$, where 2^Ω denotes the power set, which is the set of all possible subsets of Ω . Thus, $y \in \mathcal{N}(x)$, means that y is a *discrete neighbor* of x . In the context of MVP problems, all iterates are required to lie on the mesh; therefore the function \mathcal{N} must be constructed so that every discrete neighbor of the current iterate must also lie on the mesh, *i.e.* $\mathcal{N}(x_k) \subseteq M_k$ for all $k = 0, 1, \dots$. Since the categorical variables are limited to a predefined list, $\mathcal{N}(x)$ is required to be finite. Additionally, the set-valued function \mathcal{N} has the reflective property, so that $x \in \mathcal{N}(y)$ for each $y \in \mathcal{N}(x)$.

With a well-defined neighborhood function for the categorical variables, local optimality in a mixed variable domain can now be defined in terms of the discrete neighbor set.

Definition 3.5 A point $x = (x^c, x^d) \in \Omega$ is a *local minimizer* of f with respect to the set of neighbors $\mathcal{N}(x) \subset \Omega$ if there exists an $\varepsilon > 0$ such that $f(x) \leq f(v)$ for all v in the set

$$\Omega \cap \bigcup_{y \in \mathcal{N}(x)} (B(y^c, \varepsilon) \times y^d) \quad (3.4)$$

where $B(y^c, \varepsilon)$ is an open ball of radius ε centered at y^c . Because this definition applies to the local neighborhoods of all the discrete planes in the discrete neighbor set, this definition imposes a stronger condition than simply requiring optimality with respect to the discrete neighbor set and the local continuous neighborhood of the incumbent. Additionally, the quality of the local minimizer is directly related to the definition of \mathcal{N} . By increasing the number of members in the discrete neighbor set, the local minimizer becomes more global; however, the number of function evaluations needed to satisfy the optimality condition increases.

In general, optimization requires that for a point to be considered a local minimizer, a first-order necessary condition must be satisfied. As with the previous definition, this condition also requires revision in order to apply to the mixed variable domain. In doing so, $\nabla^c f$ denotes the gradient of f with respect to the continuous variables. The following definition given by Sriver [55] is presented by Lucidi *et al.* [38] in a different form:

Definition 3.6 A point $x \in \Omega$ satisfies *first-order necessary conditions* for optimality if

1. $(w^c - x^c)^T \nabla^c f(x) \geq 0$ for any feasible $(w^c, x^d) \in \Omega$;
2. $f(x) \leq f(y)$ for any discrete neighbor $y \in \mathcal{N}(x) \subset \Omega$;
3. $(w^c - y^c)^T \nabla^c f(y) \geq 0$ for any discrete neighbor $y \in \mathcal{N}(x)$ satisfying $f(y) = f(x)$ and for any feasible $(w^c, y^d) \in \Omega$.

Definition 3.6 requires stationarity at x with respect to the continuous variables (condition 1), optimality with respect to the discrete neighbors (condition 2), and stationarity with respect to the continuous variables at each discrete neighbor of x (condition 3). Finally, the definition of convergence must be revised to fit within the context of a mixed variable domain.

Definition 3.7 (Abramson [2]) Let $\Omega \subseteq (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d})$ be a mixed variable domain. A sequence $\{x_i\} \in \Omega$ is said to *converge* to $x \in \Omega$ if, for every $\varepsilon > 0$, there exists a positive integer N such that $x_i^d = x^d$ and $\|x_i^c - x^c\| < \varepsilon$ for all $i > N$.

Under the mild conditions, Audet and Dennis [5] showed that for bound constrained optimization of MVP problems, the MGPS algorithm converges to a point satisfying the first-order necessary conditions for optimality, which is extended to linear constraints by Abramson [2].

3.3 Bound and Linear Constraints

For bound and linear constraints, the barrier approach described in Section 2.4.1 is applied in conjunction with the tangent cone generator approach of Lewis and Torczon [36] across each discrete plane. In the barrier approach, infeasible points are not evaluated and their function values are set to $+\infty$. While this prevents infeasible points from replacing the incumbent, it does not guarantee convergence to a stationary point. Lewis and Torczon [36] establish that, in order for GPS to converge to a first-order stationary point of a linearly constrained problem, each direction set D^i must be sufficiently rich so that the polling directions D_k^i can be chosen to conform to the geometry of the nearby constraint boundaries. This is done by including in each D^i the tangent cone generators for every point in Ω^c . The tangent cone is defined as follows (Nocedal and Wright [46]):

Definition 3.8 A vector $w \in \mathbb{R}^n$ is *tangent* to X at $x \in X$ if, for all vector sequences $\{x_i\}$ with $x_i \rightarrow x$ and $x_i \in X$, and all positive scalar sequences $t_i \downarrow 0$, there is a sequence $w_i \rightarrow w$ such that $x_i + t_i w_i \in X$ for all i . The *tangent cone* at x is the collection of all tangent vectors to X at x .

Since linear constraints form a convex set, for $x \in X$, the tangent cone to X at x can be expressed as $T_X(x) = \text{cl}\{\mu(w - x) : \mu \geq 0, w \in X\}$.

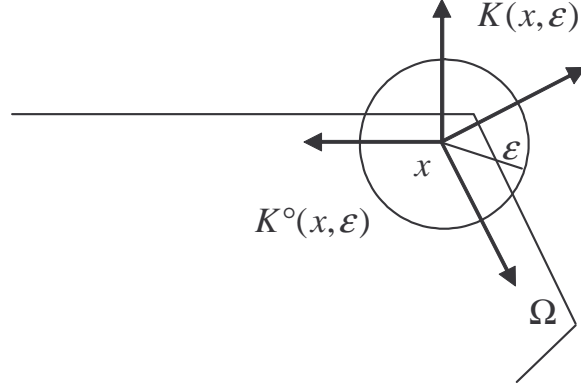


Figure 3.1. Tangent Cone Near Boundary (adapted from Lewis and Torczon [36])

As presented by Lewis and Torczon [36], algorithms which conform to the geometry of the linear constraint boundaries need to conform only to the nearby boundaries. If the current iterate is within $\varepsilon > 0$ of a constraint boundary, then the tangent cone $K^\circ(x, \varepsilon)$ can be represented as the polar of the cone $K(x, \varepsilon)$ of normals for the constraints within ε of x_k . A representative of the geometric relationship between these boundaries and cones is illustrated in Figure 3.1.

The following definition from Audet and Dennis [6] formalizes this concept.

Definition 3.9 A rule for selecting the positive spanning sets $D_k = D(k, x_k) \subseteq D$ conforms to X for some $\varepsilon > 0$, if at each iteration k and for each y in the boundary of X for which $\|y - x_k\| < \varepsilon$, $T_X(y)$ is generated by nonnegative linear combination of the columns of a subset D_k^y of D_k .

By this definition, when $x_k \in \Omega$ is not near a boundary, there are no additional requirements imposed on the positive spanning set; it is only when $x_k \in \Omega$ is near a boundary that the positive spanning set is required to contain directions that conform to the boundaries of the active constraints.

3.4 MGPS Algorithm

The mixed variable generalized pattern search (MGPS) algorithm for deterministic optimization extends GPS methods to MVP problems through modification to the polling performed on the feasible region. As noted in Section 2.4.1, the SEARCH step is not required to guarantee convergence; thus the only restriction of the SEARCH step is that it evaluates a finite number of mesh points, denoted as $S_k \subset M_k$. Since the mesh is defined for all the discrete planes of the MVP structure, it should be noted that in the MGPS algorithm the search for points of improvement is not limited to the current discrete plane.

For the treatment of categorical variables, polling in the feasible region is conducted in separate POLL and EXTENDED POLL steps. At iteration k , the POLL step evaluates points from the set $P_k(x_k)$ and from the set of discrete neighbors $\mathcal{N}(x_k)$. If the algorithm fails to find improvement from the SEARCH and POLL steps, the EXTENDED POLL step is conducted to poll around points in the set of discrete neighbors whose objective function value is sufficiently close to that of the incumbent. To identify such points, the *extended poll condition*, given by

$$f(x_k) \leq f(y) < f(x_k) + \xi_k \quad (3.5)$$

must be satisfied by a point $y \in \mathcal{N}(x_k)$ in the discrete neighbor set of the incumbent x_k , to be considered for extended polling. The ξ_k parameter, referred to as the *extended poll trigger* at iteration k , is a scalar satisfying $\xi_k \geq \xi > 0$, where ξ is a user-defined lower bound. It is typically set as a percentage of the objective function value (but bounded away from zero) (see Abramson [2]). In a manner similar to the user-defined neighbor set, the quality of the local minimizer is determined in part by the selection of the tolerance

value ξ . By increasing ξ , the search for points of improvement becomes more global, since extended polling will be triggered more frequently.

For discrete neighbors that satisfy the extended poll condition, polling is conducted in a manner similar to the POLL step of the original GPS method; that is, polling is restricted to the discrete plane of the current discrete neighbor point. By initiating the polling around a particular discrete neighbor y_k , the extended polling sequence $\{y_k^j\}_{j=1}^{J_k}$ is terminated when either $f(y_k^{J_k} + \Delta_k(d, 0)) < f(x_k)$ for some $d \in D_k(y_k^{J_k})$ or $f(x_k) \leq f(y_k^{J_k} + \Delta_k(d, 0))$ for all $d \in D_k(y_k^{J_k})$, where J_k is the total number of extended poll points considered in the EXTENDED POLL step. By this construction of extended polling, the set of extended poll points evaluated about a particular discrete neighbor y_k can be denoted as

$$\mathcal{E}(y_k) = \left\{ P_k(y_k^j) \right\}_{j=1}^{J_k} \quad (3.6)$$

regardless of the terminating condition of the polling sequence. Thus, the set of all extended poll points considered by the EXTENDED POLL step at iteration k is defined as

$$\mathcal{X}_k(\mathcal{E}_k) = \bigcup_{y \in \mathcal{N}_k^\xi} \mathcal{E}(y_k) \quad (3.7)$$

where $\mathcal{N}_k^\xi = \{y \in \mathcal{N}(x_k) : f(x_k) \leq f(y) < f(x_k) + \xi_k\}$ is the subset of discrete neighbors that trigger extended polling.

The trial set T_k that is used to conditionally update the mesh size parameter can now be expressed as the union of all the iterates generated by the SEARCH, POLL, and EXTENDED POLL steps, denoted as $T_k = S_k \cup P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\mathcal{E}_k)$. Abramson [2] uses this notation to formalize the following definitions for MVP problems:

Definition 3.10 If $f(y) < f(x_k)$ for some $y \in T_k$, then y is said to be an *improved mesh point*.

Definition 3.11 If $f(y) \geq f(x_k)$ for all $y \in P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\mathcal{E}_k)$, then y is said to be a *mesh local optimizer*.

The updating of the mesh parameter is performed in exactly the same manner as in the original GPS method, given in Figure 2.6. Specifically, when an improved mesh point is found within trial set T_k , the iteration is considered successful and the mesh parameter is updated as

$$\Delta_{k+1} = \tau^{m_k} \Delta_k \quad (3.8)$$

where $\tau > 1$ is a rational number that remains constant over all iterations, and the integer m_k satisfies $0 \leq m_k \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$. However, if the iteration is unsuccessful, then the mesh parameter is updated as

$$\Delta_{k+1} = \tau^{m_k} \Delta_k \quad (3.9)$$

where $\tau > 1$ is a rational number that remains constant over all iterations, $\tau^{m_k} \in (0, 1)$, and the integer m_k satisfies $m_{\min} \leq m_k \leq -1$ for some fixed integer m_{\min} .

The Audet and Dennis [5] MGPS algorithm for deterministic bound constrained optimization is shown in Figure 3.2. Under the mild assumptions that the level set

$$\mathcal{L}_\Omega(x_0) = \{x \in \Omega : f(x) \leq f(x_0)\} \quad (3.10)$$

is compact and the objective function f is continuously differentiable over a neighborhood of $\mathcal{L}_\Omega(x_0)$ when the discrete variables are fixed, and the rule for selecting directions conforms to Ω^ε for some $\varepsilon > 0$, then the MGPS algorithm converges to a point satisfying the first-order necessary conditions for optimality.

Mixed Variable Generalized Pattern Search (MGPS) Algorithm

Initialization: Choose a feasible starting point x_0 such that $f_\Omega(x_0) < \infty$.

Set a discrete neighbor set \mathcal{N} and extended poll trigger $\xi > 0$.

Let D be a positive spanning set.

Let the $M_0 \subset \Omega$ be the mesh defined by mesh size parameter $\Delta_0 > 0$ and $D_0 \in D$.

Set the iteration counter k to 0.

For $k = 0, 1, \dots$

1. Set extended poll trigger $\xi_k \geq \xi$.
2. SEARCH Step (Optional): Employ some finite strategy seeking an improved mesh point; *i.e.*, $x_{k+1} \in M_k$ such that $f_\Omega(x_{k+1}) < f_\Omega(x_k)$.
3. POLL Step: If the SEARCH step does not find an improved mesh point, evaluate f_Ω at the points in $P_k(x_k) \cup \mathcal{N}(x_k)$ until an improved mesh point x_{k+1} is found (or until done).
4. EXTENDED POLL step: If the SEARCH and POLL steps did not find improved mesh point, evaluate f at points in $\mathcal{X}_k(\xi_k)$ until either an improved mesh point x_{k+1} is found (or until done).
5. Update: If SEARCH, POLL or EXTENDED POLL finds an improved mesh point, then update x_{k+1} , and set $\Delta_{k+1} = \tau^{m_k} \Delta_k \geq \Delta_k$ where $\tau > 1$ is a rational number that remains constant over all iterations, and the integer m_k satisfies $0 \leq m_k \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$;
- Otherwise, set $x_{k+1} = x_k$, and set $\Delta_{k+1} = \tau^{m_k} \Delta_k < \Delta_k$ where $\tau > 1$ is a rational number that remains constant over all iterations, $\tau^{m_k} \in (0, 1)$, and the integer m_k satisfies $m_{\min} \leq m_k \leq -1$ for some fixed integer m_{\min} .
6. Terminate the algorithm if the stopping criterion is met or if the budget of function evaluations is reached; otherwise, return to step 1.

Figure 3.2. MGPS Algorithm for Deterministic Optimization (adapted from Abramson [2])

3.5 Proposed Modifications

Modifications for improving the efficiency of MGPS and extending the applicability of NOMADm to stochastic simulations are now presented. The following two subsections describe both the purpose behind and the incorporation of the modifications presented in Section 1.2.

3.5.1 Search Approach

As noted in Section 2.4.2, the SEARCH step in the GPS algorithm allows the user the flexibility to employ any finite heuristic in an attempt to quickly find improvement. While the SEARCH step contributes nothing to the convergence theory, a wise choice can greatly increase efficiency and even affect the quality (Booker *et al.* [9]).

For MVP problems, Audet and Dennis [5] retained the SEARCH step in the MGPS algorithm. As noted in Section 3.4, because the mesh is defined across all the possible categorical settings, the search can be applied globally. If the SEARCH step is unsuccessful in finding an improved mesh point, the algorithm simply proceeds to the next step. However, if the SEARCH step locates a improved mesh point, the incumbent is replaced, Δ_k is updated, and the current iteration ends. Thus, an efficient search can lead to a reduction in function evaluations to reach a local optimizer.

The need for a modification of the MGPS algorithm can be illustrated by the case in which a SEARCH step would locate a point w in a discrete plane with different categorical settings than the incumbent. Under the current algorithm, a SEARCH step is only performed prior to the POLL step; thus, a user is limited to the discrete planes on which a search heuristic may be applied. Thus, under the standard application of MGPS, the user would have to perform a complete MGPS iteration before performing the SEARCH step that would locate the point w . The proposed change to the MGPS algorithm is to include an EXTENDED SEARCH step that provides the user the ability to employ a finite search heuristic on any of the discrete planes before the EXTENDED POLL step is performed.

The EXTENDED SEARCH step can increase the efficiency of the algorithm not only by reducing the number of required iterations, but also by reducing the sampling requirements when surrogates are used. Since the EXTENDED SEARCH step occurs after the POLL step,

a user could choose to construct surrogates only for those discrete planes that indicate areas of possible improvement, *i.e.* those that correspond to discrete neighbors that trigger extended polling. Because a space-filling set of points is commonly used in constructing an initial surrogate, the EXTENDED SEARCH step provides a means for constructing surrogates only as required, without doing so for every discrete plane.

Another user option that this step introduces is the extended polling of points generated by the EXTENDED SEARCH step. Suppose that the point w does not provide strict improvement over the incumbent x_k but provides strict improvement over the discrete neighbor of the incumbent $y \in \mathcal{N}(x_k)$. Since $f(w) \geq f(x_k)$, the point w is not considered an improved mesh point and the algorithm would progress from a SEARCH step to a POLL step. However, since the objective function value of the point w indicates a possible area of improvement, the user may consider polling around the point w in addition to the discrete neighbor y during the EXTENDED POLL step. From a convergence standpoint, it is important to note that one cannot directly replace the discrete neighbor y in the EXTENDED POLL with a point in the discrete plane that provides strict improvement with respect to the discrete neighbor, such as the point w . If the last iterate of the EXTENDED POLL initiated at the point x is denoted as $z_k(x)$, then there is no guarantee that $f(z_k(w)) \leq f(z_k(y))$.

In a manner similar to the SEARCH step, the EXTENDED SEARCH step provides the user the ability to employ a finite search heuristic to accelerate the convergence of the algorithm without affecting the underlying convergence theory. As a result, the computational efficiency of the MGPS algorithm can be increased by including the EXTENDED SEARCH step without requiring new convergence results.

3.5.2 R&S Procedure

Sriver [55] showed that under mild assumptions a R&S procedure can be used in a MGPS algorithm for stochastic systems to provide almost sure convergence to an appropriately defined first-order stationary point. The assumptions can be divided into those relating to the algorithm (presented for the MGPS algorithm in Section 3.3) and those relating to the statistical properties of the problem and R&S procedure selected (provided in this section). Just as in Section 3.2, the special structure of MVP problems requires a revision of almost sure convergence to include the mixed variable domain found in Sriver’s work.

Definition 3.12 Let $\Omega \subseteq (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d})$ be a mixed variable domain. A sequence of multivariate random vectors $\{X_k\}$ converges almost surely (a.s.) to the limit point $x \in \Omega$ if, for every $\varepsilon > 0$, there exists a positive integer N such that $\Pr\{X_k^d = x^d\} = 1$ and $\Pr\{\|X_k^c - x^c\| < \varepsilon\} = 1$ for all $k > N$.

As part of numerical testing, Sriver [55] performed a comparative analysis of competing direct search methods under various R&S procedures; namely, Rinott’s two stage procedure [51], a screen-and-select procedure of Nelson *et al.* [45], and Sequential Selection with Memory (SSM) of Pichitlamken and Nelson [48]. Since SSM was reported from the comparative analysis to offer performance advantages over the other R&S procedures, it was chosen as the R&S procedure to implement in this research.

As noted in Section 2.5, SSM is a *fully sequential procedure* specifically designed for iterative search routines, in which one sample at a time is taken from every candidate still in play and eliminates clearly inferior ones as soon as their inferiority is apparent. In order to locate inferior candidates, SSM performs a pairwise statistical test at every iteration for each of the candidates. By removing the candidates that are statistically inferior to all the other members of the candidate set, SSM provides a computationally efficient procedure to

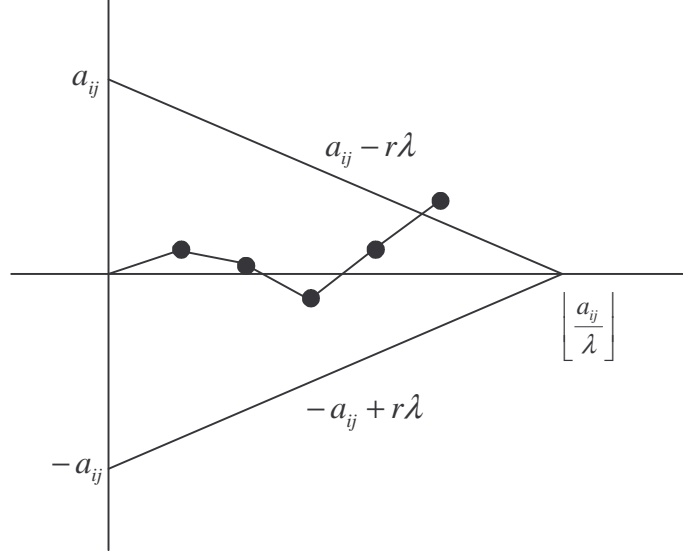


Figure 3.3. Triangular Continuation Region for SSM

select the best candidate. Another advantage of SSM is the utilization of previously stored sampling data, which alleviates some of the computational burden of obtaining additional simulation outputs at each iteration of the optimization algorithm.

The key in understanding SSM is the concept of the *triangular continuation region*, which is now detailed. It is constructed as the result of an initial stage of sampling used to estimate the variances between each pair of candidates. Using the notation of Section 2.5, with X_{ip} , $p = 1, \dots, r_{\max}$, denoting the replications of candidate X_i , for $i = 1, \dots, k$, the estimate of the variances between each pair of candidates $\sigma_{ij}^2 = \mathbb{V}[X_{ip} - X_{jp}]$ is calculated as

$$S_{ij}^2 = \frac{1}{s_0 - 1} \sum_{p=1}^{s_0} (F_{ip} - F_{jp} - [\bar{F}_i(s_0) - \bar{F}_j(s_0)])^2 \quad (3.11)$$

where i and $j = 1, \dots, k$, s_0 is the user-defined number of initial replications and

$$\bar{F}_i(s_0) = \frac{1}{s_0} \sum_{p=1}^{s_0} X_{ip}.$$

The resulting variance estimates, together with the user-defined indifference zone parameter δ and level of significance parameter α , establish the parameters of the triangular continuation area; namely,

$$\lambda = \frac{\delta}{2c} \quad \text{and} \quad a_{ij} = \frac{\eta(s_0 - 1)S_{ij}^2}{4(\delta - \lambda)} \quad (3.12)$$

where the recommended values for $c \in \mathbb{R}$ and $\eta \in \mathbb{R}$ are 1 and $((k - 1)/(2\alpha))^{2/(s_0 - 1) - 1}$, respectively. The general triangular continuation region for the SSM procedure is illustrated in Figure 3.3, where r represents the number of samples of the SSM procedure.

Recall from Section 2.5 that Hartmann [25] replaced the large deviation bound by a Brownian motion bound on the acceptance region. Let $B(t; \delta, \sigma^2)$ denote the Brownian motion process with $\mathbb{E}[B(t; \delta, \sigma^2)] = \delta t$ and $\mathbb{V}[B(t; \delta, \sigma^2)] = \sigma^2 t$. Hartmann [25] demonstrated that the Brownian motion process $B(t; \delta, \sigma^2)$ with $\delta > 0, m > 0, \lambda > 0$, and stopping time defined as

$$T = \inf\{t : |B(t; \delta, \sigma^2)| \geq \lambda(m - t)\} \quad (3.13)$$

exits toward the lower boundary of the continuation region with probability

$$Pr\{B(t; \delta, \sigma^2) < 0\} = \int_{-\infty}^{\infty} \frac{e^{-2\lambda\varepsilon/\sigma}}{1 + e^{-2\lambda\varepsilon/\sigma}} \phi\left(\frac{\varepsilon - (m\delta/\sigma)}{\sqrt{m}}\right) \frac{d\varepsilon}{\sqrt{m}} \quad (3.14)$$

Since the triangular region defines an incorrect selection as breaking the lower bound of the triangular region, the total probability that the SSM procedure selects the wrong candidate is given by

$$\sum_{i=1}^{k-1} \mathbb{E}[Pr\{B(t; \delta, \sigma_{ik}^2) < 0\}] \quad (3.15)$$

Fabian [22] showed that when $\lambda = \delta/2c$ and $m = a_{ik}/\lambda$, Equation (3.15) is equivalent to

$$\sum_{i=1}^{k-1} \sum_{\ell=1}^c (-1)^{\ell+1} [1 - \frac{1}{2} I(\ell = c) \exp\{\frac{(2c-\ell)}{2c-1} (\frac{-2a_{ik}}{\sigma_{ik}^2})(\delta - \lambda)\}] \quad (3.16)$$

where I is the indicator function.

By substitution of the recommended values for λ and a_{ij} from Equation (3.12) and under the assumption that the candidates are normally distributed, the expression in Equation (3.16) is equivalent to α , which produces the desired level of significance and indifference given by Equation (2.21).

Using this construction of the triangular continuation region, the maximization procedure of SSM (see Pichitlamken and Nelson [49]) can be converted into the required minimization selection procedure for this research. The resulting procedure, using the recommended values for c and η , is given in Figure 3.4.

As part of the convergence analysis for the MGPS-RS algorithm, Sriver [55] demonstrated that by using a MGPS algorithm, almost sure convergence for stochastic systems is guaranteed by enforcing the following assumptions:

- A1:** All iterates X_k produced by the MGPS algorithm lie in a compact set.
- A2:** The objective function f is continuously differentiable with respect to the continuous variables.
- A3:** For each set of discrete variables X^d , the corresponding set of directions $D^i = G_i Z_i$, as defined in (3.1), includes tangent cone generators for every point in X^c .
- A4:** The rule for selecting directions D_k^i conforms to X^c for some $\varepsilon > 0$ (see Definition 3.9).

Sequential Selection with Memory Procedure

Initialization: Choose a number of stage zero samples $s_0 \geq 2$.

Let s_{ic} denote the number of previously stored responses of candidate X_i .

For any member $X_i \in \{X_1, \dots, X_k\}$ with a stored responses $s_{ic} < s_0$, collect $s_0 - s_{ic}$ more responses.

Choose a significance level $\frac{1}{k} < 1 - \alpha < 1$ and indifference zone parameter δ .

1. Estimate $\sigma_{ij}^2 = \mathbb{V}[X_{ip} - X_{jp}]$ with

$$S_{ij}^2 = \frac{1}{s_0 - 1} \sum_{p=1}^{s_0} (F_{ip} - F_{jp} - [\bar{F}_i(s_0) - \bar{F}_j(s_0)])^2$$

2. Compute the procedure parameters as

$$\lambda = \frac{\delta}{2} \text{ and } a_{ij} = \frac{\eta(s_0 - 1)S_{ij}^2}{4(\delta - \lambda)}$$

where $\eta = ((k - 1)/(2\alpha))^{2/(\underline{s}_{ij}-1)-1}$.

3. Let $N_{ij} = \lfloor \frac{a_{ij}}{\lambda} \rfloor$, $N_i = \max_{j \neq i} \{N_{ij}\}$, $N = \max_{1 \leq i \leq k} N_i$. If $s_0 > N$, then stop and select the candidate with the smallest sample mean as the best. Otherwise, let $I = \{1, \dots, k\}$ be the set of surviving solutions, set counter $r = s_0$ and proceed to Step 4.

4. Set $I^{\text{old}} = I$. Let

$$I = \left\{ i : i \in I^{\text{old}} \text{ and } R_i \leq \min_{i \in I^{\text{old}}, j \neq i} (R_j + a_{ij}) - \frac{r\delta}{2} \right\}$$

where

$$R_j = \begin{cases} \sum_{p=1}^r X_{jp}, & \text{if } s_{ic} < r \\ \frac{r}{s_{jc}} (\sum_{p=1}^{s_{jc}} X_{jp}), & \text{otherwise} \end{cases}$$

5. If $|I| = 1$, then stop and report the only survivor as the best; otherwise, for each candidate $X_i \in \{i \in I : s_{ic} < r + 1\}$ collect one additional response sample and set $r = r + 1$. If $r = N + 1$, terminate the procedure and select the solution in I with the smallest sample mean as the best; otherwise, for each $i \in I$ go to Step 4.

Figure 3.4. Sequential Selection with Memory (adapted from Pichitlamken and Nelson [48])

A5: For each $i = 1, 2, \dots, k$, the responses $\{F_{ip}\}_{p=1}^{r_{\max}}$ are independent, identically and normally distributed random variables with mean $f(X_i)$ and unknown variance $\sigma_i^2 < \infty$, where $\sigma_\ell^2 \neq \sigma_q^2$ whenever $\ell \neq q$.

A6: The sequence of significance levels $\{\alpha_r\}$ satisfies $\sum_{r=0}^{\infty} \alpha_r < \infty$, and the sequence of indifference zone parameters $\{\delta_r\}$ satisfies $\lim_{r \rightarrow \infty} \delta_r = 0$.

A7: For the r th R&S procedure, the probability of correctly selecting the best candidate $X_{[1]}$ is at least $1 - \alpha_r$ whenever $f(Y_{[i]}) - f(Y_{[1]}) \geq \delta_r$ for any $i \in \{2, 3, \dots, k\}$.

A8: For all but a finite number of MGPS iterations and sub-iterations, the best solution $X_{[1]}$ is unique; *i.e.*, $f(X_{[1]}) \neq f(X_{[i]})$ for all $i \in \{2, 3, \dots, k\}$, where each member of the candidate set lies on the mesh defined at iteration k .

Assumption **A1** is a standard assumption of the MGPS algorithm. A simplifying assumption on the target class of problems for this research is represented by **A2**. Assumptions **A3** and **A4** are required for convergence of the MGPS algorithm. Assumption **A5** is a common requirement for R&S techniques and can be achieved in simulation by batching the output data or the use of sample averages of independent replications Nelson *et al.* [45]. Assumption **A6** is a requirement imposed by the convergence theory presented by Srivier [55]. By requiring the R&S parameters to decay, the incumbents are forced to provide almost sure convergence to a limit point. Assumption **A7** enforces the iterative correct selection guarantee of the R&S procedure from Equation (2.21). Finally, assumption **A8** is required to ensure that the indifference zone condition is eventually met during the course of the iteration sequence.

Since the proposed modification to the MGPS algorithm does not alter the underlying convergence theory, implementation of the EXTENDED SEARCH step does not violate assumptions **A1-A4**. Additionally, by placing an update step within the modified algorithm, in a similar manner as Srivier's [55] MGPS-RS algorithm, assumption **A6** is maintained. The remaining assumptions **A5**, **A7**, and **A8** can be satisfied by proper selection of the simulation model and R&S procedure. Therefore, the modified MGPS algorithm, presented in Figure 3.5, provides almost sure convergence to a first-order stationary point

Modified Mixed Variable Generalized Pattern Search Algorithm

Initialization: Choose a feasible starting point x_0 such that $f_\Omega(x_0) < \infty$.

Set a discrete neighbor set \mathcal{N} and extended poll trigger $\xi > 0$.

Let D be a positive spanning set.

Let the $M_0 \subset \Omega$ be the mesh defined by mesh size parameter $\Delta_0 > 0$ and $D_0 \in D$.

Set the R&S parameters $\alpha_0 \in (0, 1)$, and $\delta_0 > 0$.

Set the iteration counter k to 0.

Set the R&S counter r to 0.

For $k = 0, 1, \dots$

1. Set extended poll trigger $\xi_k \geq \xi$.

2. SEARCH Step (Optional): Using an R&S procedure with parameters α_r and δ_r for candidate evaluation, employ some finite strategy seeking an improved mesh point.

Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$.

3. POLL Step: If the SEARCH step does not find an improved mesh point, evaluate f_Ω at the points in $P_k(x_k) \cup \mathcal{N}(x_k)$, using an R&S procedure with parameters α_r and δ_r , until an improved mesh point x_{k+1} is found (or until done).

Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$.

4. EXTENDED SEARCH Step (Optional): Using an R&S procedure with parameters α_r and δ_r for candidate evaluation, employ some finite strategy seeking an improved mesh point.

Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$.

5. EXTENDED POLL step: If the SEARCH, POLL, and EXTENDED SEARCH steps did not find improved mesh point, evaluate f at points in $\mathcal{X}_k(\xi_k)$, using an R&S procedure with parameters α_r and δ_r , until either an improved mesh point x_{k+1} is found (or until done).

Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$.

6. Update: If SEARCH, POLL or EXTENDED POLL finds an improved mesh point, then update x_{k+1} , and set $\Delta_{k+1} = \tau^{m_k} \Delta_k \geq \Delta_k$ where $\tau > 1$ is a rational number that remains constant over all iterations, and the integer m_k satisfies $0 \leq m_k \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$;

Otherwise, set $x_{k+1} = x_k$, and set $\Delta_{k+1} = \tau^{m_k} \Delta_k < \Delta_k$ where $\tau > 1$ is a rational number that remains constant over all iterations, $\tau^{m_k} \in (0, 1)$, and the integer m_k satisfies $m_{\min} \leq m_k \leq -1$ for some fixed integer m_{\min} .

7. Terminate the algorithm if the stopping criterion is met or if the budget of function evaluations is reached; otherwise, return to step 1.

Figure 3.5. Modified MGPS Algorithm for Simulation Optimization

for stochastic simulation systems, and converges to a first-order stationary point for deterministic simulation systems.

3.6 Implementation Considerations

With the modified MGPS-RS algorithm explained, issues concerning the implementation of the modified code into NOMADm may now be discussed. Although SSM allows for the re-use of historical data of all previously sampled iterates in future R&S procedures, in this research only the data for the current incumbent is maintained. Since NOMADm was originally developed for deterministic simulations, data corresponding to previously sampled iterates are maintained in a cache structure to prevent re-sampling of the same point, thus reducing the overall number of required function evaluations. However, the randomness in stochastic problems makes it at least possible that a previously evaluated trial point is, in truth, a better point, and would be discovered to be so with more replications. If the R&S parameters are not relatively small enough when an iterate is initially sampled, the data stored in the cache of NOMADm may be inaccurate. Since the costs of storing complete iterate data can be prohibitive, the objective function values of stochastic simulation responses are estimated and entered in the cache as the sample mean; thus the estimate can be inaccurate unless R&S parameters are relatively small. Then in future samplings, when the R&S parameters are tight enough to produce an accurate estimate, the iterate will never be re-sampled or eligible to replace the incumbent point since it is already in the cache. This issue is partially remedied by the assumption that the initial point is not too close to the optimal and by the fact that the mesh is refined over time, thus allowing for the possibility of future samplings that are near the cached point to be evaluated accurately. Although the incumbent's sample mean is entered into the cache, the sample data for the incumbent is maintained outside of the cache. Once the incumbent is replaced, the historical data is updated by the new incumbent. The reason that this exception is made

for the incumbent is that the incumbent is the most frequently compared point and thus this is expected to produce the largest computational savings.

The use of an EXTENDED SEARCH step also introduces an issue when considering modification to the NOMADm code. Since the original SEARCH was performed before the POLL step, NOMADm was designed to only sample on the local mesh of the incumbent. Although the EXTENDED SEARCH allows for a more global search, sections of the SEARCH routine had to be changed to avoid the creation of a surrogate across different categorical settings. Specifically, at each iteration, the cache is now filtered to include only those iterates that have the same categorical variable values as the incumbent. Additionally, since the number of function evaluations needed to create a surrogate for every discrete neighbor can quickly exceed the number of function evaluations saved by the use of the surrogate, a surrogate is only built when a discrete neighbor point triggers the extended polling condition (see Equation (3.5)). This avoids the building of surrogates for every possible discrete neighbor, as is done by Srivier [55], and is more general than the original MGPS algorithm.

3.7 Summary

This chapter detailed the requirements for the MGPS algorithm to converge to a point satisfying the first-order necessary condition in the context of a mixed variable domain. Additionally, the modified algorithm with the sequential selection with memory procedure was presented. In the next chapter, the efficiency of using the modified algorithm within the NOMADm is examined through a computational evaluation.

Chapter 4 - Computational Evaluation

In this chapter, the proposed optimization algorithm is implemented on test problems to characterize its applicability to general engineering design problems. The surrogate approaches that are used in the testing are described in order to explain their selection and purpose. Then, the design of the experimental investigation is presented, including the test problem and series of runs performed. Finally, the results are analyzed with a focus on the computational efficiency of the modified algorithm.

4.1 Research Surrogates

As mentioned in Section 2.5, an inexpensive surrogate constructed by interpolating or smoothing known values can improve the efficiency of the optimization algorithm by reducing the number of function evaluations required to locate a point that produces a decrease. In order to limit the number of additional function evaluations, the surrogates used in this research are constructed as approximations of the true objective function and are based on previously computed responses that are stored by NOMADm in a cache. The resulting surface is then used in the SEARCH step to select candidate points that actively seek regions of improved mesh points.

Although the use of a surrogate requires an investment of initial function evaluations and each SEARCH step requires additional function evaluations, if the surrogate can successfully locate areas of improvement, then the surrogate may accelerate the convergence of the algorithm to a stationary point. As a result, the use of surrogates may reduce the overall number of function evaluations for an optimization algorithm to produce an appropriate solution. To increase the likelihood that the surrogate is able to find such areas of improvement, and thus contribute positively to the optimization algorithm, surrogates are traditionally selected to incorporate some knowledge of the underlying system. Since

the responses in this optimization problem are assumed to be generated by a "black box" system, the selection of surrogates is limited by the general information that the assumptions of the problem type provide. In particular, the objective function is assumed to be continuously differentiable; thus this research uses surrogates that assume the underlying approximated surface is smooth. Two general methods that use this smoothness assumption are the nonparametric approach of kernel regression and the parametric approach of Kriging.

4.1.1 Kernel Regression

As presented by Härdle [24], the basic idea of smoothing of a dataset $\{(X_i, Y_i)\}_{i=1}^n$ involves the approximation of the mean response curve f in the regression relationship

$$Y_i = f(X_i) + \varepsilon_i \quad (4.1)$$

where $i = 1, \dots, n$ and in the context of this research, X_i are design vectors, Y_i are the averaged responses of the associated X_i vector, and ε_i is random noise of the system. The underlying assumption is that, if f is smooth, then an observation X_i near x should contain information about the value of f at x . Kernel regression is a means of quantifying this linear relationship. The approximation of the mean response curve f is traditionally calculated by the Nadaraya-Watson estimator ([42] and [62]) that uses a weighted sequence based on a kernel. For the two dimensional case, the Nadaraya-Watson estimator is given as

$$\hat{f}(x) = \frac{\sum_{i=1}^n K_h(x - X_i) Y_i}{\sum_{i=1}^n K_h(x - X_i)} \quad (4.2)$$

where $K_h(u) = h^{-1}K(u/h)$ is the kernel with bandwidth parameter $h > 0$. The *kernel* is defined as a continuous, bounded and symmetric real function K which integrates to unity (Härdle [24]); that is,

$$\int_{-\infty}^{+\infty} K(u)du = 1 \quad (4.3)$$

The kernel defines the shape of the surface between known response points. In this research, it is assumed that the decay from a known response is normally distributed; thus, the Gaussian kernel, $K(u) = \exp(-u^2/2)/\sqrt{2\pi}$, is used. The *bandwidth* of the kernel represents how much weight is placed on the interpolation with respect to the distance from the known response points. Once the kernel is chosen, the estimator can be re-written in a form that illustrates the dependence of the estimator function on the bandwidth. For this research, the resulting estimator function can be expressed as

$$\hat{f}(x_j, h) = \frac{\sum_{\ell=1, \ell \neq j}^N \bar{F}_\ell \exp\left(-\frac{D_\ell^2}{2h^2}\right)}{\sum_{\ell=1, \ell \neq j}^N \exp\left(-\frac{D_\ell^2}{2h^2}\right)} \quad (4.4)$$

where $D_j^2 = \|x - x_j\|_2^2$ represents the squared Euclidean distance from x to x_j . As Srivier noted [55], the bandwidth h essentially determines the degree of nonlinearity in the surrogate function. As h increases, the curvature in \hat{f} decreases such that, when h is very large, \hat{f} is a constant that assumes the mean value of all \bar{F}_j ; *i.e.*, $\hat{f} \rightarrow \frac{1}{N} \sum_{j=1}^N \bar{F}_j$ as $h \rightarrow \infty$. Smaller values of h allow more curvature in \hat{f} but can cause outliers to have too great an effect on the estimate. As $h \rightarrow 0$, \hat{f} assumes the value of \bar{F}_j for the corresponding x_j that is nearest the estimation point.

Since the accuracy of the estimator depends mainly on the bandwidth parameter h , several bandwidth selection procedures have been introduced (see Härdle [24]). The one employed in this research is the *leave-one-out cross-validation* method, in which the sum of squares error (SSE)

$$SSE(h) = \sum_{j=1}^N (\hat{f}(x_j, h) - \bar{F}_j)^2 \quad (4.5)$$

is calculated by omitting, in turn, each design vector x_j in Equation (4.4). The resulting function SSE can then be optimized separately to determine an appropriate bandwidth setting for the parameter h .

4.1.2 Kriging

Similar to kernel regression, Kriging uses the smoothness assumption of the objective function to estimate responses under the context that the closer the inputs are, the more positively correlated are their outputs (van Beers and Kleijnen [61]). Although the overall assumption of the surface is the same, the generation of estimation functions is quite different. Not only is Kriging a parametric method, but it is also an *exact interpolator*, in that it produces predicted values at observed input values that are exactly equal to the simulated output values.

Although Kriging was originally developed as an analysis method for mining applications (see Krige [32]), it has become a popular multidisciplinary method of estimation (see Currin *et al.* [15]). Not only have Kriging interpolation functions been shown to provide better fitting models in multi-dimensional domains than polynomial interpolants, they often exhibit fewer oscillations, which can cause polynomial fitting techniques to fail.

As in kernel regression, Kriging involves the approximation of the mean response curve f in the regression relationship

$$Y(X_i) = f(X_i) + \varepsilon(X_i) \quad (4.6)$$

where $i = 1, \dots, n$, the X_i are design vectors, Y_i are the averaged responses of the associated X_i , and ε is the random error. However, while kernel regression treats each error term as an independent variable, Kriging considers each error to be a dependent random variable with a known distribution and zero mean. By assuming the random error follows a known distribution, Kriging is able to model the covariance of the error function. The covariance is modeled as a second-order stationary process; *i.e.*, the means and variances are constants and the covariances of the outputs depend only on the distance between the inputs. For example, a general covariance function can be given as

$$Cov[f(X_i), f(X_j)] = \sigma_f^2 \exp(-\theta \|x_i - x_j\|) \quad (4.7)$$

where σ_f^2 is a scalar process variance, i and $j = 1, \dots, n$, with $i \neq j$, and θ is a weighting factor for the influence of surrounding data points.

Because Kriging is based on a regression model, Equation (4.6) can be more explicitly expressed as

$$Y(X_i) = \sum_{j=1}^k \beta_j f_j(X_i) + \varepsilon(X_i), \quad i = 1, \dots, n \quad (4.8)$$

where f_j are basis functions and β_j are the corresponding coefficients. The use of basis functions allows Kriging to assume various polynomial forms which provide more flexibility than the default linear function. Thus, if information on the underlying response surface shape is known, then the Kriging model can be tailored to take advantage of this additional knowledge.

Although Kriging allows great flexibility in the selection of the estimation function, the model parameters have to be determined before the Kriging model can be used to produce response estimates. Similar to kernel regression, these parameters are also optimized separately.

4.2 *Surrogate Sampling Design*

To garner useful information from the surrogate models, a structured approach must be taken in the selection of points which are used in the building of the surrogates. The following subsections discuss the methods employed in this research to ensure the surrogates properly balance the benefit of the additional information provided with the increase of function evaluations incurred by the optimization algorithm.

4.2.1 *Latin Hypercube Sampling*

In order to provide a good global search of the feasible region, surrogate models require response data over the entire feasible region to properly approximate the underlying surface. A simple and popular experimental design technique to provide such data is *Latin hypercube sampling* (LHS). Although LHS was invented for deterministic simulation models (Mckay *et al.* [40]), it is a useful tool for generating a space-filling set of points from which to construct an initial surrogate.

In traditional LHS, each variable domain of the feasible region is divided into n intervals of equal length. Then, design points are randomly placed within each interval so that, for each variable, a design point appears exactly once in each interval. This is convenient for pattern search, since the feasible region is already equally divided by the mesh, and thus the design points must simply be distributed properly. If the number of desired design points p equals the number of intervals n of the LHS design, then the design is said to be of *strength* one.

Since the global quality of the surrogate is initially determined by the LHS, designs of strength two ($p = 2n$) are used in this research to provide an initial search of the feasible region that is denser than a traditional strength one design ($p = n$). Figure 4.1 illustrates LHS samples of strengths one and two for a two-dimensional design space. Since locating and moving to areas of possible improvement is important early on in the algorithm, the first SEARCH step performed in this research will always be LHS.

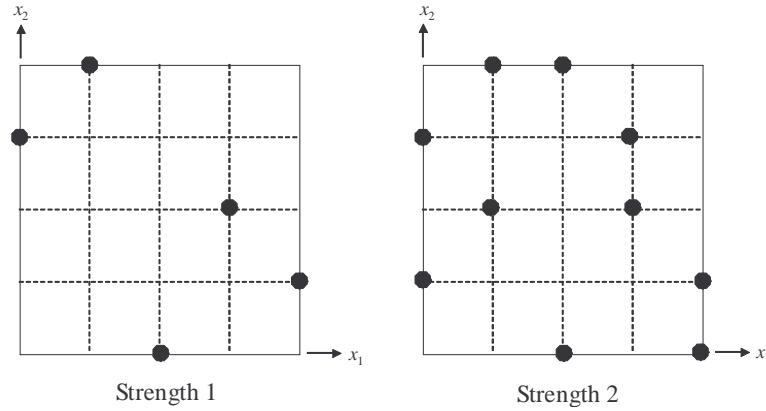


Figure 4.1. Examples of Latin Hypercube Samples of Strengths 1 and 2.

4.2.2 The Merit Function

Once the surrogate function is built, it can be utilized within the pattern search framework as an inexpensive means to generate candidate points from the mesh M_k . After the candidate points are evaluated, the points are added as design sites for the surrogate to enhance the accuracy of the surrogate function. A straightforward rule for selecting candidate points is simply to minimize \hat{f} on the mesh directly. However, using this greedy approach may inhibit the overall accuracy of the surrogate function because the trial points tend to cluster in a particular region of the design space (Srивer [55]).

Instead of directly estimating the response by the estimator function $\hat{f}(x)$, Sriver incorporated a bi-objective merit function (see Torczon and Trosset [59]) of the form

$$m(x) = \hat{f}(x) - \lambda d(x) \quad (4.9)$$

where $d(x) = \min_j \|x - x_j\|$ is the distance from x to the nearest previously sampled design site and $\lambda \geq 0$ determines the relative weight placed on the space-filling objective for the selection of candidate points. By initially setting the λ parameter as a multiple of the mean difference between initial design responses and decaying the parameter after each SEARCH step, the *merit* function approach forces the searches to perform an early examination of space-filling points. The impact of the merit function approach will be examined in computational testing.

4.2.3 The Trust Region

An additional embellishment on the standard MGPS-RS algorithm that Sriver [55] performed was the use of a trust region of the surrogate model. Since kernel regression methods are interpolatory, estimates of points lying outside the sampling region approach a value equal to the mean response of the nearest design site. Thus, the combination of a kernel regression estimator with a large bandwidth and the merit function approach would constantly force the selection of points that were distant from the current design points.

To prevent this occurrence, Sriver incorporated a "trust region" and set its radius to one-half of the maximum Euclidean distance between any pair of initial design sites. During the SEARCH step, the search is restricted to a ball centered at the starting point with the prescribed radius. The impact of this option on the quality of the surrogate will also be examined in computational testing.

4.3 *Implementation Considerations*

In addition to the general considerations presented in Section 3.6, the use of NOMADm as an implementation of the modified algorithm introduces specific encoding considerations. This section discusses the specific issues presented by and resulting decisions made from the use of NOMADm.

In order to allow the user flexibility during the local neighborhood search, NOMADm offers a wide range of direction sets, including an option of a user-defined set, to be used in the generation of the POLL set. This research considered the use of the standard maximal set, given by $[I; -I]$, to ensure a rich local search was performed.

As noted by Dennis and Schnabel [19], an important consideration in many engineering problems is the issue of variable scaling. Scaling refers to the transformation of variables so that they will have approximately the same range. For badly scaled problems, the disparate ranges of the variables have the effect of assigning unequal weighting factors to the problem variables. In extreme cases, the altering of problem variable importance can cause variables to be virtually ignored by the optimization problem. In order to maintain the relative importance of variables, NOMADm provides the option of logarithmic scaling of directions, rather than direct variable transformations. By calculating the weighting factors for each variable, the directions of the POLL set can be properly rescaled within each variable range. The result of logarithmic rescaling is that the descent directions will take into consideration the ranges of the variables. In this research, a base-2 logarithm transformation was used since it will rescale the variables more frequently than a base-10 logarithm.

In the presence of constraints, consideration must be given to the evaluation of infeasible points. Since NOMADm was designed to handle nonlinear constraints through the

use of a filter, the code already determines if potential iterates will lie within the defined constraints. Since infeasible points may improve the accuracy of the surrogate, infeasible points were evaluated but not allowed to replace the incumbent. Therefore, when the POLL set contains directions that would make a constraint active, directions which conform to the boundary were added to the POLL set and the entire POLL set was evaluated. Additionally, when stochastic simulations are used, the user-defined number of initial replications, s_0 from Equation (3.11), of infeasible points was taken. Since the goal of the optimization is to locate an improved feasible solution, the user-defined minimal number of replications was used for infeasible points to provide some variance reduction while preserving function evaluations for the overall algorithm.

When surrogates are used during the SEARCH step, the user must decide which iterates should be included in the construction of the surrogate. Although the inclusion of more (or all) design points may improve the accuracy of the surrogate, constructing and/or evaluating a surrogate with a large number of design points can become prohibitive in terms of computing speed or available memory. In an effort to balance these two concerns, only search points and improved mesh points were used during the surrogate construction; all other previously sampled points are ignored.

Finally, for mixed variable problems, MGPS typically searches only those sections of the mesh associated with the categorical setting containing the incumbent. However, the flexibility of the SEARCH step allows the search of any section of the mesh. For example, for each possible categorical setting, a surrogate can be built during initialization in preparation for future searches, as in Srivier’s implementation [55]. Since this may require a significant number of function evaluations for MVPs with many possible categorical setting combinations, an alternative approach is to build surrogates only as they are needed.

In this research, the result of the POLL step is used in determining which of the discrete neighbors are to be polled. The surrogates are only constructed for sections of the mesh that are associated with the categorical settings of discrete neighbors that trigger extended polling. Thus, surrogates are constructed only for the sections of the mesh in which there is evidence that an improved mesh point may lie.

4.4 Design of Investigation

Sequential Selection with Memory and the modified MGPS algorithm were implemented in NOMADm to extend the code to stochastic simulation problems and to improve the computational efficiency of the underlying algorithm. A computational evaluation was conducted using the new code to assess the performance of the various combinations of surrogate strategies presented in this chapter. This evaluation consisted of a series of experiments applied to three standardized test problems from Schittkowski [52].

Since the modified algorithm is a generalization of the MGPS-RS and the convergence theory is not changed, the focus of the computational testing is the use of standardized tests to compare surrogate designs. The objective of this comparative testing is to determine if, under similar termination criteria, the use of surrogates can be shown to provide a reduction in the number of function evaluations and an improved solution, where the quality of a solution is measured by its proximity to the optimal solution and the true response value at the terminating point. In order to ensure that the comparative results were based on non-confounded factors, a full factorial design was used in this testing where the factors evaluated included the type of surrogate (either Nadaraya-Watson or Kriging), the use of a merit function, and the use of a trust region. After encoding and running the deterministic test problems, stochastic versions were constructed for testing by imbuing the response with a noise signal that follows a standard normal distribution, which can be represented as

$$F(x) = f(x) + N(0, 1) \quad (4.10)$$

where $F(x)$ is the response of the simulation and $f(x)$ is the noise-free response of the system.

In order to represent a cross-section of standard test problems, the following three nonlinear unconstrained problems were selected.

Problem 1 (Powell function)

General Polynomial: $f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$

with an initial point $x_0 = (3, -1, 0, 1)^T$,

initial objective function value $f(x_0) = 235$,

optimal point $x^* = (0, 0, 0, 0)^T$,

and optimal objective function value $f(x^*) = 0$

Problem 2

Quadratic: $f(x) = x^T Q x$ where $Q_{ij} = \frac{1}{i+j-1}$, $i, j = 1, 2, 3, 4$ (4×4 Hilbert Matrix)

with an initial point: $x = (-4, -2, -1.333, -1)^T$,

initial objective function value $f(x_0) = 33.965$,

optimal point $x^* = (0, 0, 0, 0)^T$,

and optimal objective function value $f(x^*) = 0$

Problem 3 (Rosenbrock function)

Sum of Squares: $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 100(x_3 - x_2^2)^2 + (1 - x_2)^2$

$+100(x_4 - x_3^2)^2 + (1 - x_3)^2$

with an initial point: $x = (-1.2, 1, -1.2, 1)^T$,

initial objective function value $f(x_0) = 532.4$,

optimal point $x^* = (1, 1, 1, 1)^T$,

and optimal objective function value $f(x^*) = 0$

4.4.1 Deterministic Runs

To characterize the general impact the use of the selected surrogates would have on the test problems, the results of using no surrogates were compared to each of the surrogates of this study (Nadaraya-Watson and Kriging estimators) without the use of a trust region or merit function approach. For each of the runs, the noise was removed from the response so that any difference found would be attributable to the surrogate used. Although the solution becomes deterministic in the no-surrogate case, the use of the initial LHS procedure produces randomness in the results when surrogates are employed. Thus, in addition to the use of a terminating criteria of either 5000 function evaluations or $\Delta_k \leq 0.0001$, 100 replications of each test problems were taken whenever the Nadaraya-Watson or Kriging estimators were used.

The results of these test runs show a mild improvement when surrogates are used. Specifically, reductions were observed in the distance from the true minimizer and in true objective function value of the final iteration. The number of function evaluations was also lower when surrogates were used. Replications corresponding to surrogate runs with a number of function evaluations less than the median are correlated with runs that provide reductions in the normed distance and true function evaluation. Therefore, the surrogates provide an overall improvement when used on these test problems. Details are provided in Appendix A.1.

4.4.2 Pilot Study

Since the optimization algorithm is dependent on appropriately selected R&S parameters to produce useful estimates of the true surface response, a pilot study was performed on each of the problems to determine the R&S parameters used in the main test runs. The

initial estimates for the proper R&S parameters are taken from the work of Sriver in [55]. For the pilot study, the number of initial samples was fixed at 5 and the decay rate for both the alpha and indifference were varied identically starting at 0.95. The other parameters investigated for change were the initial alpha and indifference zone parameters which were 0.8 and 100, respectively. The pilot study considered alternative settings for alpha of 0.70, indifference zone of 80, and decay parameters of 0.90.

By conducting a pilot study, common R&S parameter settings for the problems tested could be found that provide an appropriate baseline convergence to the known optimal point, where the baseline refers to the implementation without surrogates. Establishing a baseline, the relative worth of using a particular surrogate can be directly judged in comparative analysis with the runs without surrogates. Thus, the full factorial pilot study was run without the use of surrogates.

Using one hundred replications of each design setting, the quality of the R&S parameter settings was judged by the distance from the terminating solution to the true known optimum. One should note that the quality of the solution was not judged by the resulting response at the terminating solution as the response itself is stochastic and is related to the distance to the optimum. The pilot study used terminating criteria of either 5000 function evaluations or $\Delta_k \leq 0.0001$. Since the underlying algorithm produced stochastic results at each iteration, shown in Appendix A.2, the resulting distribution of the terminating solution was of an unknown analytic form.

The pilot run results were initially analyzed by graphical tools to provide a basic characterization of the underlying probability distributions. Since most standard statistical tools for comparing distributions rely on underlying normal distributions, an initial characterization of the distributional form is important in justifying the use of parametric tools.

If the distributions are not normal or cannot justifiably be transformed to a normal distribution, then the distribution must be compared using nonparametric tools, many of which have the same power as parametric methods. The outlier box plot and quantile box plot of the distributions shown in Appendix A.2 provide sufficient evidence that the distributions are not normal; thus, nonparametric methods were used.

For the pilot study results, the van der Waerden test was performed. This test is used to detect whether at least two of k sample populations come from different distributions (see Sheskin [53]). By transforming the rank-orders into a set of standard deviations derived from the standard normal distribution, the van der Waerden test provides statistical power generally equal to that of the analogous parametric test. The null hypothesis is that the samples are derived from the same underlying distribution, and its rejection, based on the given level of significance, indicates that there is enough statistical evidence to conclude that the different R&S parameters produce different underlying distributions. Thus, even if transformations were used to normalize the distributions, the R&S parameters impact the quality of the terminating solution. Additionally, the quality of the parameter settings was measured by the median, as opposed to the mean. Because each distribution may contain extreme values, as indicated by the outlier box plots, the mean of the distribution can be skewed. Since the median is not influenced by extreme observations, it was chosen as a more appropriate measure of central tendency.

The results of the pilot runs, shown in Appendix A.2, reference the R&S parameter design by concatenating the indifference zone, the indifference rate of decay, the alpha level, and the alpha rate of decay. Thus, the R&S settings suggested by Srivier can be represented as 100958095 for an indifference zone of 100, alpha level of 0.80, and decay rates of 0.95. From the results, R&S parameter settings of 100958095, 100908090, and 75907090 tended

to produce favorable distance results. Therefore, these initial conditions settings were used in the computational testing of the main run.

4.4.3 Main Runs

Since the purpose of the computational testing is to find if, starting under similar conditions, the use of surrogates can be shown to provide an improved solution, the baseline of no surrogates is accompanied by each of the surrogate design combinations for the main run. The quality of solution is judged by both the proximity to the optimal solution and the true objective function value at termination, where the main run used terminating criteria of either 100,000 function evaluations or $\Delta_k \leq 0.0001$. Just as in the pilot run, box plots of the distributions are used to initially characterize each of the distributions. The comparative analysis is then performed both on the grouped results of all three R&S parameter settings and for each R&S parameter individually. When the box plots for two or more distributions indicate that there may be similarity between the distributions, either the Wilcoxon rank-sum procedure or the Kruskal-Wallis procedure is performed. The Wilcoxon rank-sum procedure tests the hypothesis that the two sample populations are different and the Kruskal-Wallis procedure can be considered an extension of the Wilcoxon procedure that tests whether at least two of k sample populations are different. These procedures test the null hypothesis of equal distributions through the comparison of transformed rank values for each distribution. Again, the median is used in measuring the overall quality of the solution for each of the distributions.

4.5 Computational Results

The results of the main runs are shown in Appendix A.3, where the abbreviations NW and Krig are used to identify the use of the Nadaraya-Watson and Kriging surrogates and the use of a local trust region and a merit function are represented by a boolean indicator;

e.g., KrigLoc1Mer0 represents the use of a Kriging surrogate with a local trust region but not a merit function. Unlike the deterministic runs, the results indicate that the use of surrogates in the present implementation does not provide improvement in convergence to a local optimum and the number of function evaluations does not indicate that the surrogates would provide an overall improvement to the terminating solution. In fact, the number of function evaluations for the surrogate runs exceeded the no-surrogate runs without any indication of an improved response. However, the reason for this failure and the differences in competing surrogate designs' performances provide enough information to warrant some general discussion.

4.5.1 Main Run Results

At first, failure of the surrogates to provide an improved response, as demonstrated by the deterministic runs, was unexpected. Since the surrogates actively use prior information from the cache to generate trial points, it was assumed that the surrogate runs would demonstrate an improved terminating solution and a reduction in the number of function evaluations. However, the use of decaying R&S parameters have the affect of improving the cache estimates over time and, other than the incumbent, are never updated after the initial sampling. Thus, the surrogates are autocorrelated with their occurrence within the cache. As such, the surrogate is constructed from points with various levels of confidence due to the different sampling variances. The resulting surrogate is such a poor fit for the true surface that the generation of trial points decreases the computational efficiency of the algorithm.

The results also indicate that the use of a trust region and a merit function can impede the algorithmic progress to a stationary point when there is a single optimum. The surrogate design using the Kriging estimator with a combined trust region and merit function

approach (KrigLoc1Mer1) provided the worst solution across each of the test problems, and the surrogate design using the Nadaraya-Watson estimator with a combined trust region and merit function approach (NWLoc1Mer1) frequently performed worse than the other Nadaraya-Watson based designs. Since the use of a merit function forces the search to become more space-filling, the associated test runs perform a more global search than other test runs. When multiple local optima are present, the purpose of the optimization may be to find an area that simply improves the current solution; thus a global search may be appropriate. However, in this research, each of the test functions contained a single global optimum. By rigorously searching the feasible region, the combined trust region and merit function approach requires additional function evaluations to ensure alternative areas of improvement were not overlooked during the convergence to a stationary point. Thus, when compared to test runs that perform more local searches, searches that used the combined trust region and merit function approach performed poorly. However, if the test problems had contained many local optima, it is likely that the combined approach would have performed well.

Additionally, there appears to be a strong correlation between competing sets of surrogate designs. In each of the test problems, when a local trust region was not used, the Nadaraya-Watson estimator performed similarly regardless if the merit function was employed or not. However, when the Kriging estimator was employed, the terminating solutions for designs where a merit function was not employed were similar to each other, regardless if the local merit region was used. Since this similarity was apparent from the box plots, both of these cases were statistically tested for equivalence of underlying distributional form. As shown in Appendix A.3, there is not enough statistical evidence to reject the null hypothesis of equivalent underlying distributions for these surrogate designs.

These observations demonstrate the interpolatory nature of each of these surrogates. Since the Gaussian kernel used for the Nadaraya-Watson has long tails, the estimates outside the sampling region may have been tending to distant points. Also, since the optimal was centrally located within the domain of the problem, the Kriging estimator may have ensured that a local trust region was maintained by modeling the covariance as Gaussian distribution.

Computational testing demonstrated that using a cache of mean responses, calculated from previously evaluated iterate responses, can have a large influence on the quality of the terminating solution and the number of function evaluations. As a general result of this research, in cases where the quality of iterate response estimation improves as the algorithm progresses, the use of previous iterate responses from a cache can negatively impact the performance and terminating solution of an optimization algorithm. In this research, the use of previously evaluated iterates impacts both the R&S procedure used for solving the iterate selection subproblem and the quality of the surrogate used for locating areas of improved response. To demonstrate that the use of the cache negatively impacted the algorithm, additional testing was performed by modifying the R&S procedure as discussed in the following section.

4.5.2 R&S Modification Results

The unmodified R&S procedure used for the main runs maintained previously stored values for the current incumbent for future R&S comparisons. Since the quality of iterate estimation improves asymptotically, an initial bias of low functional responses are built into the cache of the incumbent. If the initial estimates for the incumbent are significantly below the true response value, the incumbent may not be replaced by an iterate whose true response is lower than the incumbent. Thus, the algorithm is likely to produce poor termi-

minating solutions. In order to address the issue, the memory portion (the storage of previous response evaluations for the incumbent) was removed from the R&S procedure and the main runs were redone with the modified code. The removal of the incumbent memory for the R&S procedure enables more accurate response estimates of the incumbent by requiring it to be re-estimated with the current R&S parameter setting. Thus, response estimation of the incumbent is improved by requiring additional function evaluations. From the comparison shown in Appendix A.4, the removal of former incumbent response data improved both observed and true responses for the terminating solution. Since the underlying algorithm is dependent on accurate incumbent information, bias in the sample estimation of the incumbent not only reduces the accuracy of the incumbent response, but also reduces the quality of the terminating solution. By using the modified R&S procedure, the terminating solutions, shown in Appendix A.5, demonstrate improvement over the unmodified R&S procedure results of Appendix A.3. Therefore, the initial savings in functional evaluations provided by the storage of the incumbent value has been shown to be an ineffective means of improving the terminating solution for stochastic responses for the optimization algorithm used in this research.

The use of previously evaluated iterates also impacts the quality of the surrogate used during a search. Although the surrogates are constructed during each SEARCH step, which has the affect of recalibrating the surrogate prior to its use, the design iterates used to build the surrogate are not recalibrated to ensure homogeneity of the sample variances. As noted in Section 3.6, since the R&S parameters are initially set loosely and are decayed as the algorithm progresses, responses selected from the cache may be inaccurate and can have large differences in sample variances. Thus, the surrogate may provide a poor fit to the true underlying surface. As a result, the mild improvement through the use

of surrogates suggested by the deterministic test problems runs is never obtained by the stochastic versions of the test problems.

4.6 Summary

This chapter demonstrated the applicability of the optimization algorithm presented in this research under various surrogate designs by comparing the quality of terminating solutions and number of function evaluations on a set of standard test problems. By analyzing the results of the computational runs, some general conclusions about the experiment design and the modified algorithm could be made. Issues raised by the analysis are discussed in the next chapter in the context of future work

Chapter 5 - Conclusions and Recommendations

This research effort, relating a generalized pattern search for mixed variable domains and ranking and selection procedures, provides an algorithm that can efficiently solve simulation-based optimization problems of "black box" systems. Through modification of the underlying MGPS algorithm, NOMADm provides enhanced support of surrogates and is now applicable to the optimization of mixed variable stochastic simulation problems with linear constraints.

5.1 Future Research

Comparative testing performed in this research has shown the relative importance of the R&S parameters, the selection of surrogates, and the surrogate approaches. As with many research efforts, the observations open the door for more discoveries in the use of surrogates, with and without R&S procedures, for simulation-based optimization. The following sections present recommendations for further research that could serve to enhance the performance of mixed variable programming problem solution techniques.

5.1.1 Nonlinear Constraints

As noted in Section 4.3, the sampling rule for infeasible points is based simply on establishing a minimal level of confidence on the response. Since infeasible points are used as design points for the construction of a surrogate, a moderate level of confidence should be maintained for the responses, which the use of minimal sampling may not provide. However, NOMADm was originally designed for optimization of nonlinearly constrained mixed variable problems through a filter approach. Since the estimation of responses are directly used as a filter criteria, a new method for controlling the sampling error of infeasible iterates will have to be considered in order to appropriately apply NOMADm to nonlinear problems.

For filter algorithms, the goal is to minimize two functions, the objective f and a continuous aggregate constraint violation function h , where $h(x) \geq 0$ if and only if x is feasible. A filter is a set of non-dominated points with respect to f and h (see Audet and Dennis [8]), and steps are considered successful whenever improvement in either function is found. By polling around the least infeasible points in the filter or the best feasible point, the goal is to improve either the least infeasible point or the best feasible point. If the minimizer lies on the boundary of the feasible region, a subsequence of iterates that approach the minimizer can be generated without requiring the polling directions to conform to the boundary.

The filter approach can be employed quickly for deterministic simulations since the parameters for each iterate, whether feasible or infeasible, are known. However, the use of a filter for stochastic simulation requires a modification to the general approach because the objective function is only available stochastically and is typically reduced by repeated sampling. A possible approach for controlling the sampling error is the use of the current R&S parameters of the feasible iterates to drive the sampling requirement for infeasible points. By the assumption of normality in the error terms, the current alpha level used in the R&S procedures for the feasible points could be used to determine the number of replications required of the infeasible points to establish a given level of significance. Additionally, the handling of stochastic constraints might be addressed by the use of a filter.

Another approach to the handling of nonlinear constraints is to extend the MADS algorithm to stochastic MVP problems. As noted in Section 2.4.2, MADS is a generalization of GPS for handling nonlinear constraints that provides stronger convergence results than MGPS. It does so by generating a dense set of mesh directions in the limit. Poll directions

are chosen at each iteration from a ever increasing set of mesh directions. The development of such a class of algorithms would allow the solution of a previously unsolvable class of problem.

5.1.2 R&S Modifications

As part of the convergence theory used in MGPS-RS, Srivier [55] demonstrated that the use of decay parameters in the R&S procedure results in asymptotic behavior of the algorithm, where the statistical significance of the selected iterate improves as the algorithm progresses. Therefore, the quality of visited solutions varies as the algorithm progresses. Since surrogates are built from past responses, the use of a nonhomogeneous pool of solutions can lead to a poor approximation of the true response surface. A method for improving the surrogates is to resample on a subset of previously visited design points and replicate the points until the desired level of homogeneity is achieved. The resampling could be performed after a fixed number of R&S procedures (which is equivalent to setting fixed alpha level thresholds) or based on the pooled sample variance of the stored responses (resampling based on the individual versus pooled variance).

Another area of current research involves balancing the cost of sampling with switching. R&S procedures that use fully sequential sampling, such as the one used in this research, require repeated switching between different model scenarios. Since the computational overhead of switching may be expensive, new methods are being developed (see Hong and Nelson [27]) that reduce the number of required switches while maintaining the required level of significance.

5.1.3 Surrogate Modifications

As shown in Equation (4.6), the Kriging model can be tailored by adding additional problem knowledge to the selection of basis functions. In this research the basis functions

were set to be linear functions, since no assumptions, other than smoothness, of the true surface were made. In true engineering problems however, additional knowledge of the problem is usually available. Through the use of alternative basis functions, the Kriging model should be able to more accurately approximate the surface. An area of future research is to develop a Kriging model that can change as the pattern search algorithm progresses as additional information is acquired.

5.2 Summary

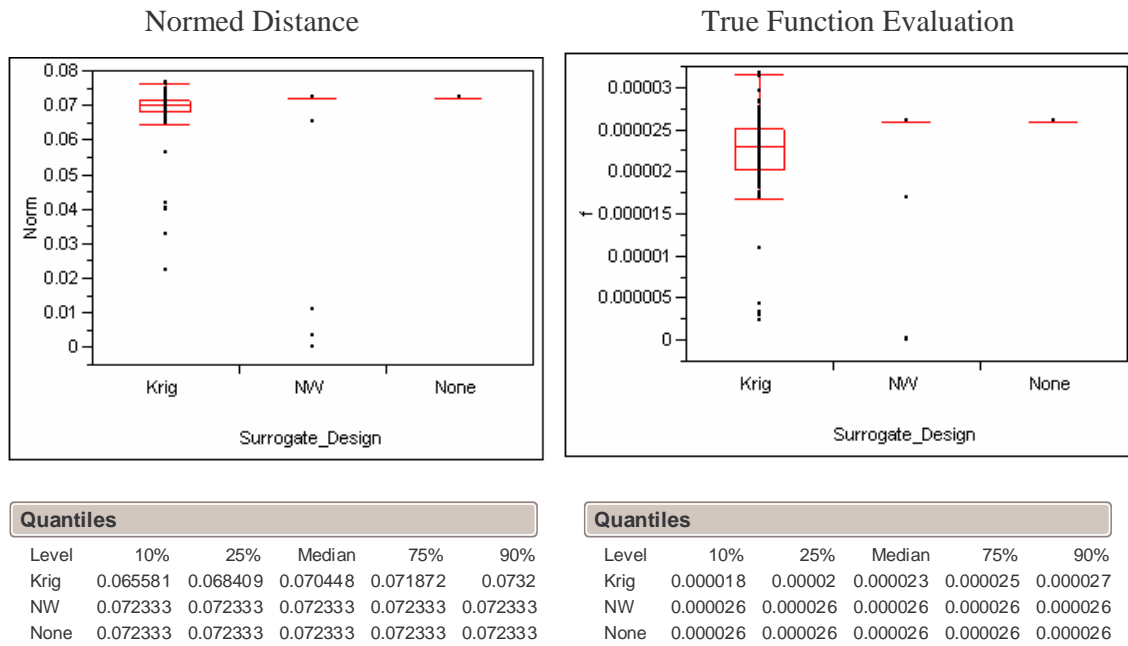
Generalized pattern search provides an effective method to solve very difficult optimization problems, *e.g.* mixed variable programming. This research has addressed the computational efficiency for optimizing simulated systems. The modifications to the mixed variable generalized pattern search were described and implemented within the NOMADm software package. The modified algorithm was tested using three stochastic test problems. Overall results were unfavorable due to the poor fit of the surrogates and lead to sample variance reduction as a suggestion for future work.

APPENDIX A - Supporting Data Summary

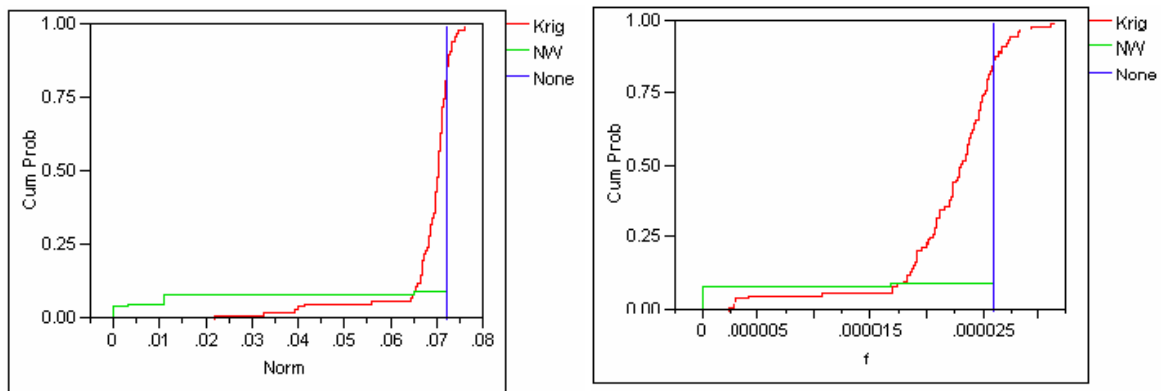
A.1 Deterministic Run Data Summary

This appendix provides the details of the initial deterministic response comparison used to establish the relative quality of solution that surrogates provide when there is no noise in the response signal. For each of the test problems, the box plot and quantiles for the performance measurements are presented.

Test Problem 1

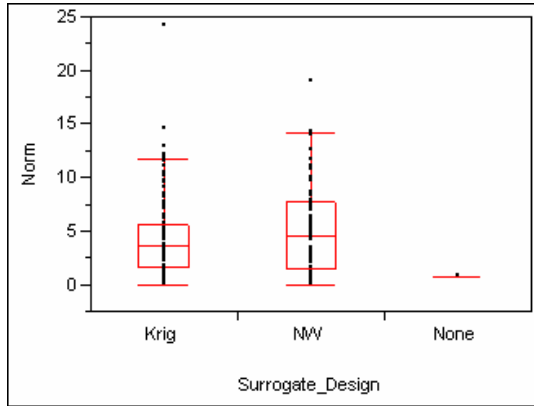


Cumulative Distribution Functions

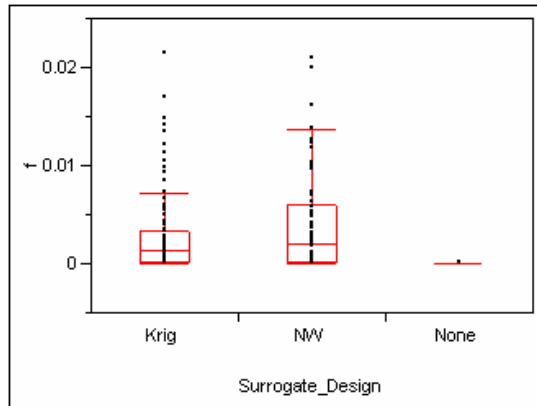


Test Problem 2

Normed Distance



True Function Evaluation



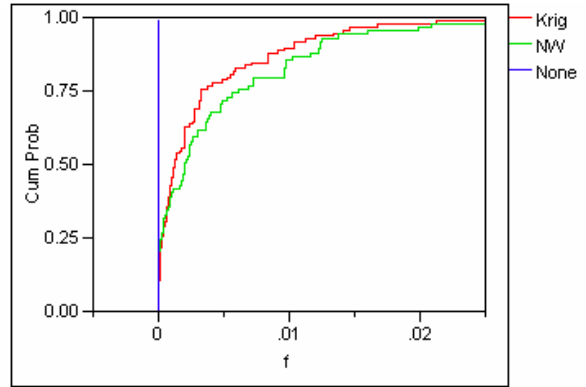
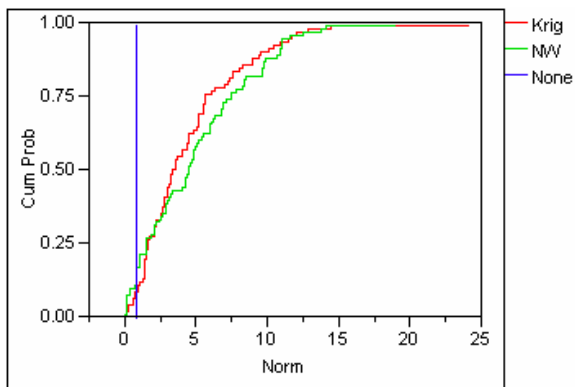
Quantiles

Level	10%	25%	Median	75%	90%
Krig	0.983785	1.737309	3.65214	5.745399	10.06253
NW	0.500934	1.599949	4.562427	7.771205	11.02117
None	0.839932	0.839932	0.839932	0.839932	0.839932

Quantiles

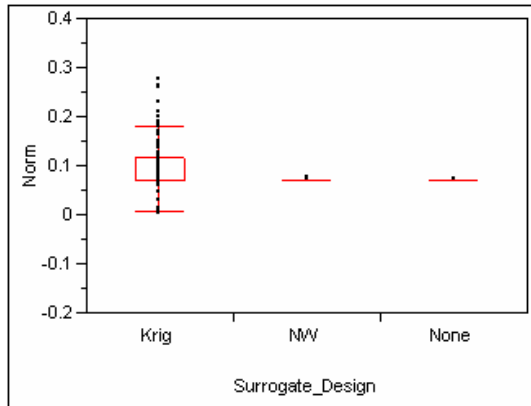
Level	10%	25%	Median	75%	90%
Krig	0.000099	0.000303	0.001352	0.00337	0.010395
NW	0.000026	0.000248	0.002095	0.006105	0.012419
None	0.000072	0.000072	0.000072	0.000072	0.000072

Cumulative Distribution Functions

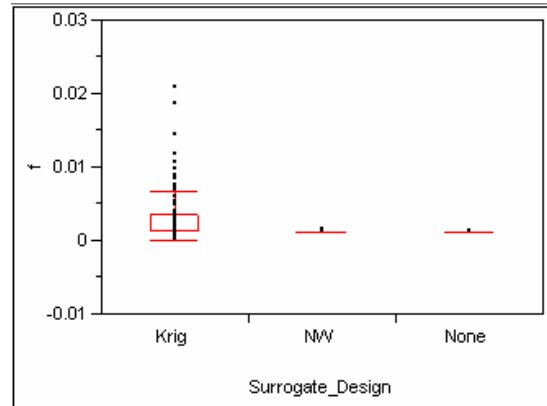


Test Problem 3

Normed Distance



True Function Evaluation



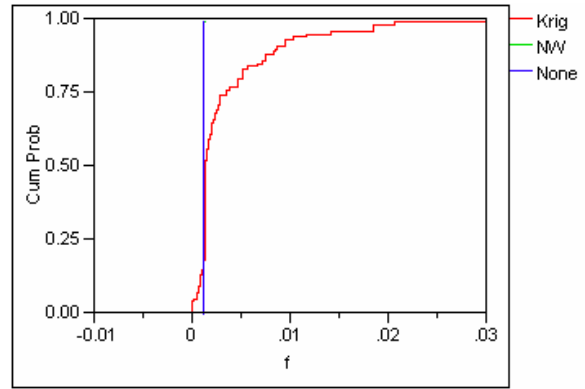
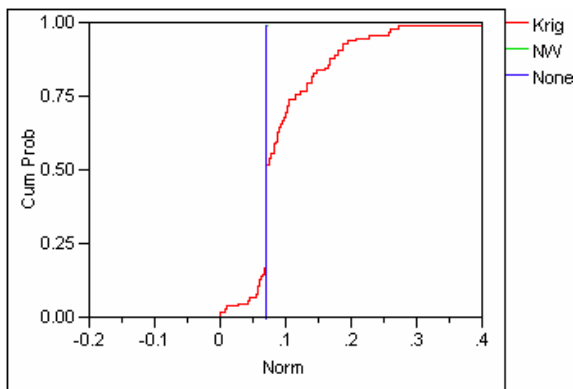
Quantiles

Level	10%	25%	Median	75%	90%
Krig	0.058642	0.072403	0.072403	0.117245	0.182248
NW	0.072102	0.072102	0.072102	0.072102	0.072102
None	0.072102	0.072102	0.072102	0.072102	0.072102

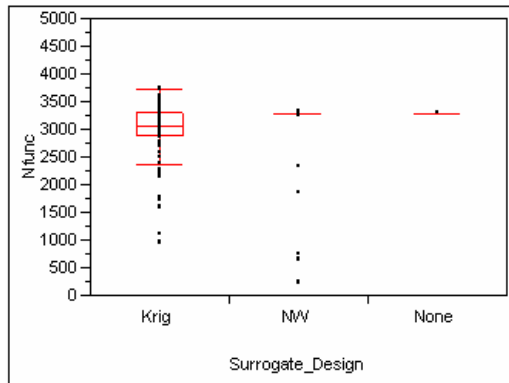
Quantiles

Level	10%	25%	Median	75%	90%
Krig	0.00087	0.001339	0.001339	0.003587	0.008814
NW	0.001324	0.001324	0.001324	0.001324	0.001324
None	0.001324	0.001324	0.001324	0.001324	0.001324

Cumulative Distribution Functions



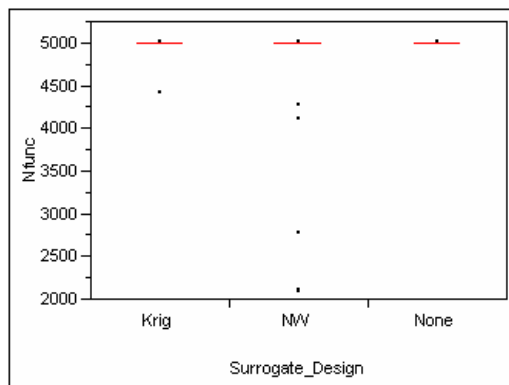
Number of Function Evaluations



Test Problem 1

Quantiles

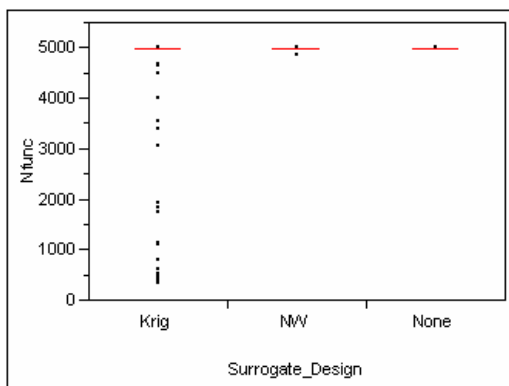
Level	10%	25%	Median	75%	90%
Krig	2230	2901	3079	3311.75	3442.9
NW	3247	3297	3297	3298	3298
None	3285	3285	3285	3285	3285



Test Problem 2

Quantiles

Level	10%	25%	Median	75%	90%
Krig	5000	5001	5002.5	5005	5006
NW	5000	5001	5003	5005	5006
None	5000	5000	5000	5000	5000



Test Problem 3

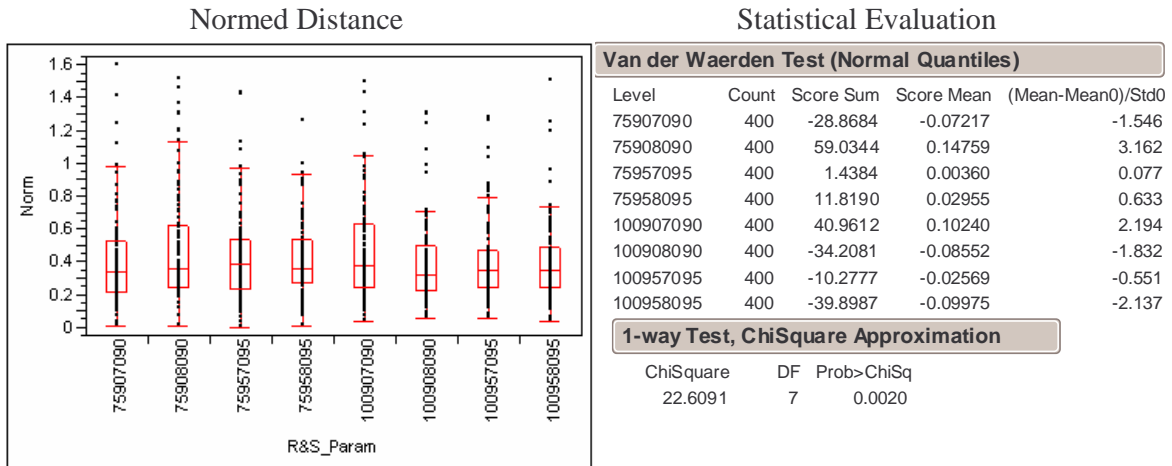
Quantiles

Level	10%	25%	Median	75%	90%
Krig	802.6	5000	5004	5005	5005
NW	5001	5001	5001	5002	5002
None	5005	5005	5005	5005	5005

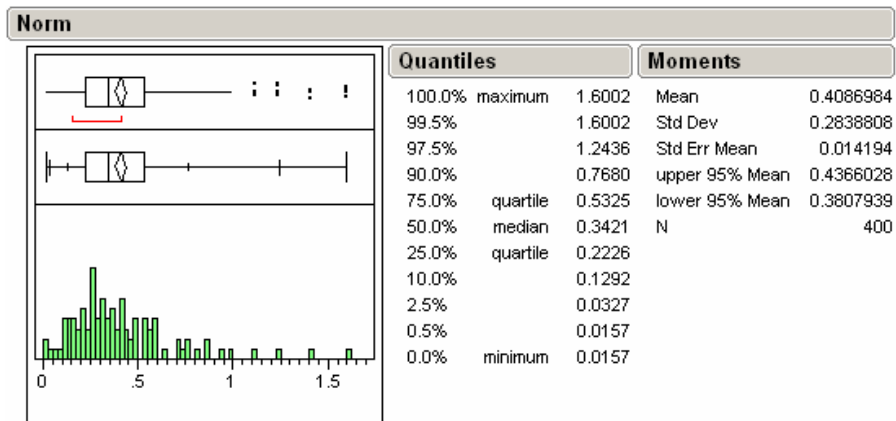
A.2 Pilot Study Data Summary

This appendix provides the details of the results of the pilot study used to establish the ranking and selection parameters for the main computational runs. As noted in Section 4.4, the distance to the optimal point was used as the quality measure for each of the R&S parameter settings. For each of the test problems, the box plot and statistical test are first presented and supported by the following individual R&S parameter histograms and descriptive statistics. The labeling of each R&S procedure design is provided by the concatenation of the indifference zone parameter, the indifference zone decay rate, the alpha level, and the alpha decay parameter, *e.g.* 100957095 indicates an initial indifference zone of 100, alpha level of 0.70, and common decay parameters of 0.95.

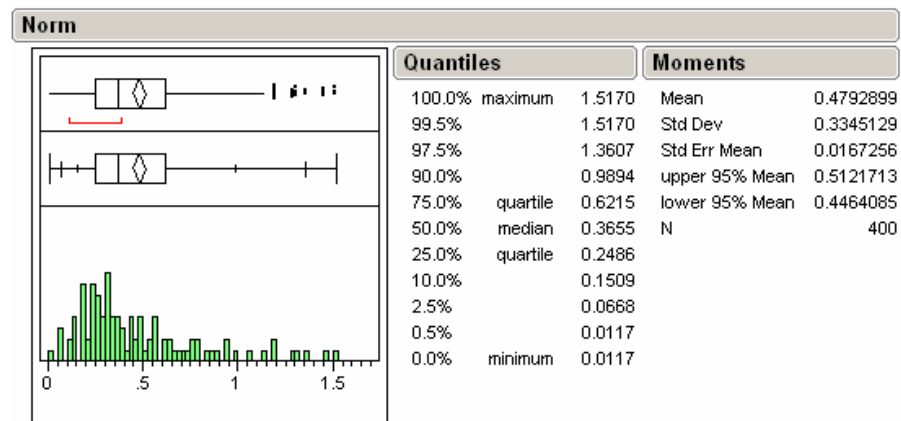
Test Problem 1



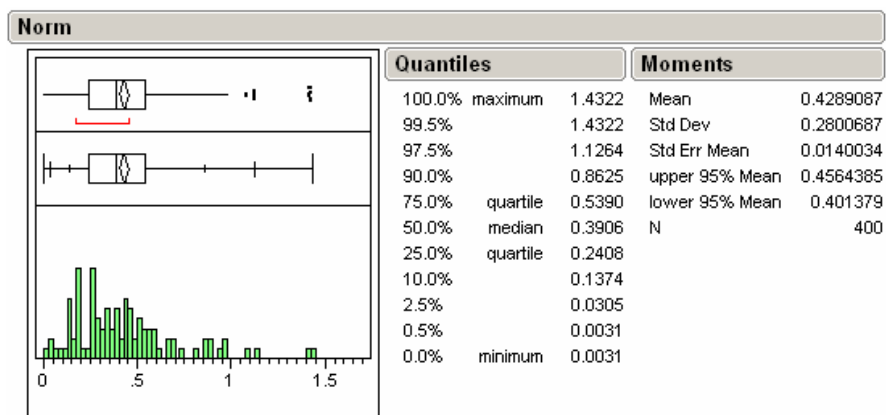
R&S Design 75907090



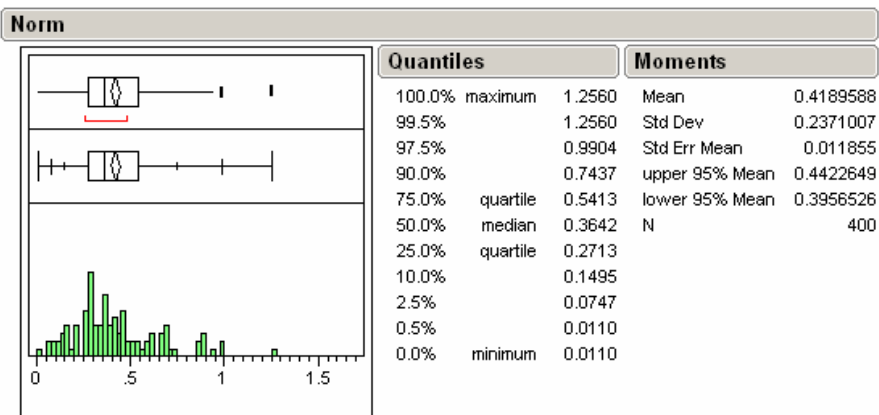
R&S Design 75908090



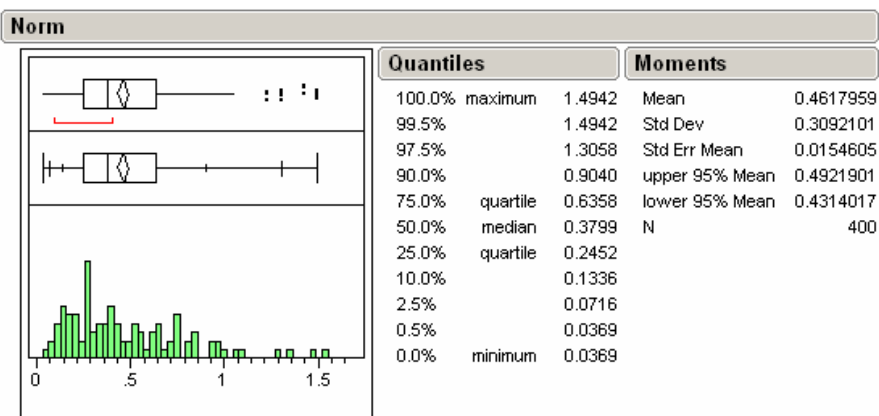
R&S Design 75957095



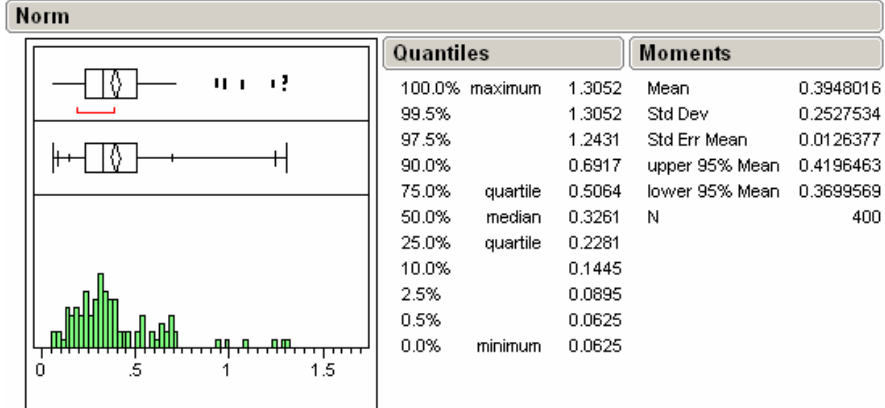
R&S Design 75958095



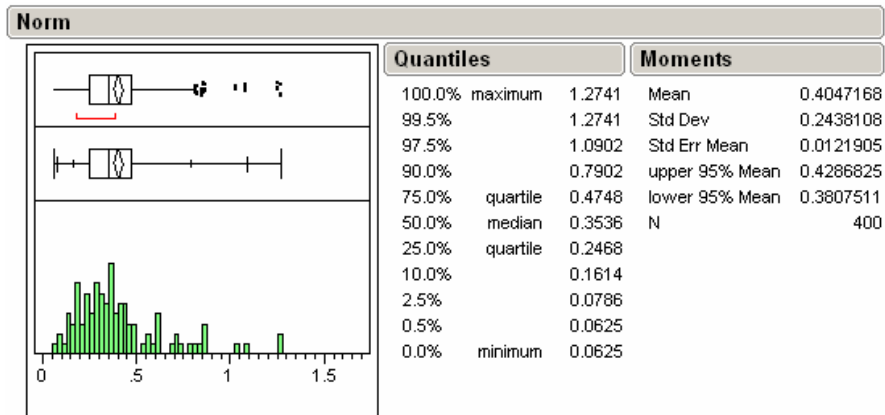
R&S Design 100907090



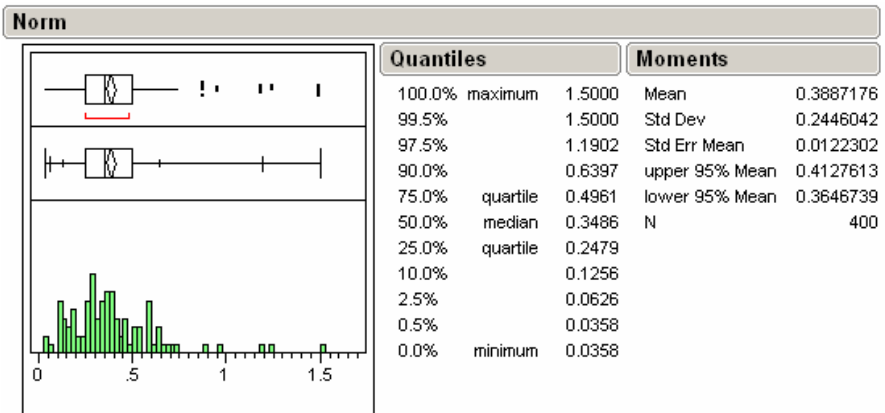
R&S Design 100908090



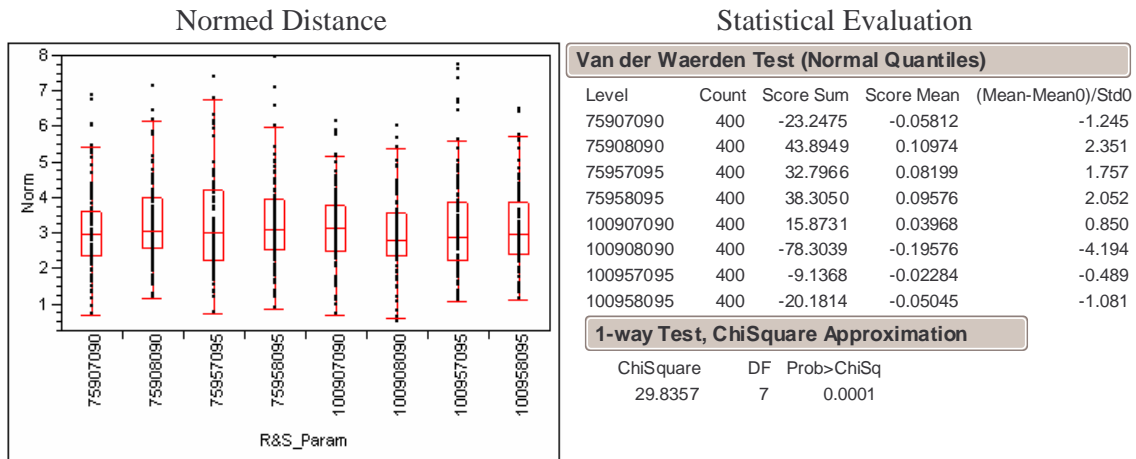
R&S Design 100957095



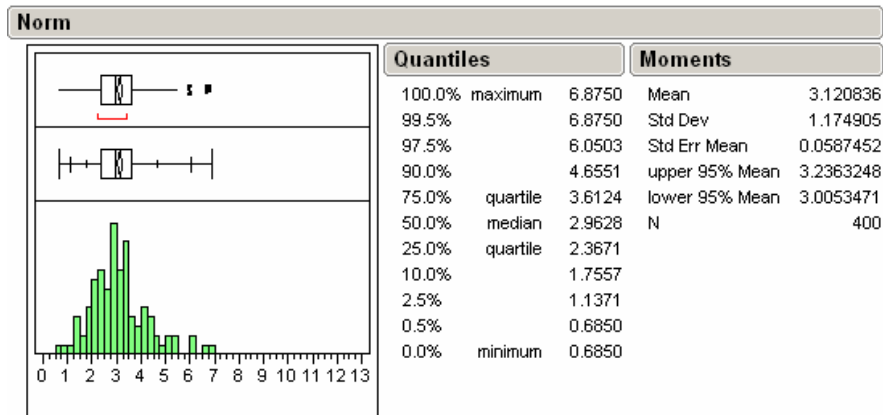
R&S Design 100958095



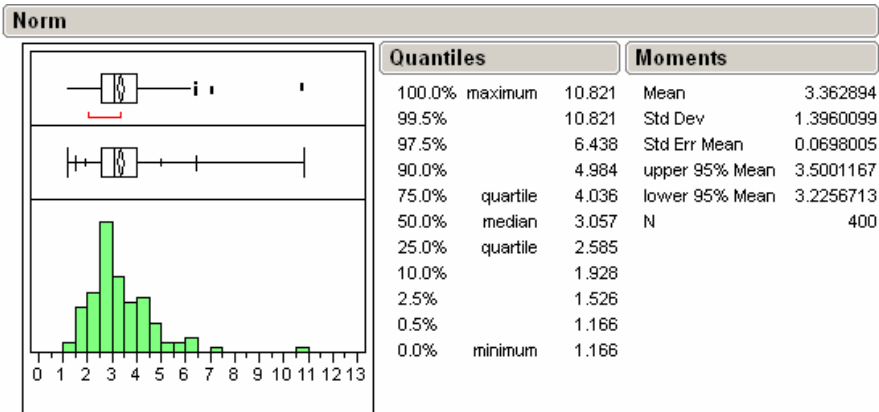
Test Problem 2



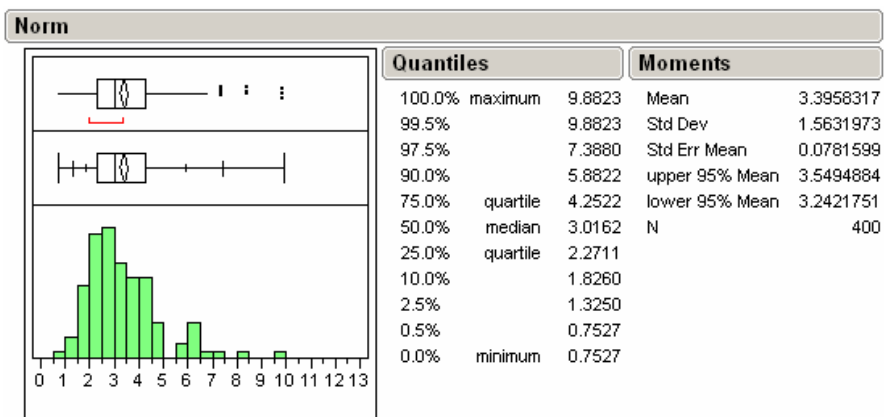
R&S Design 75907090



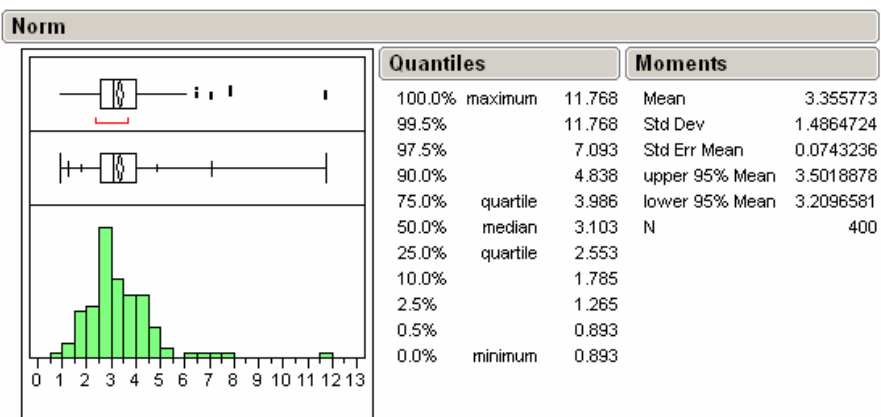
R&S Design 75908090



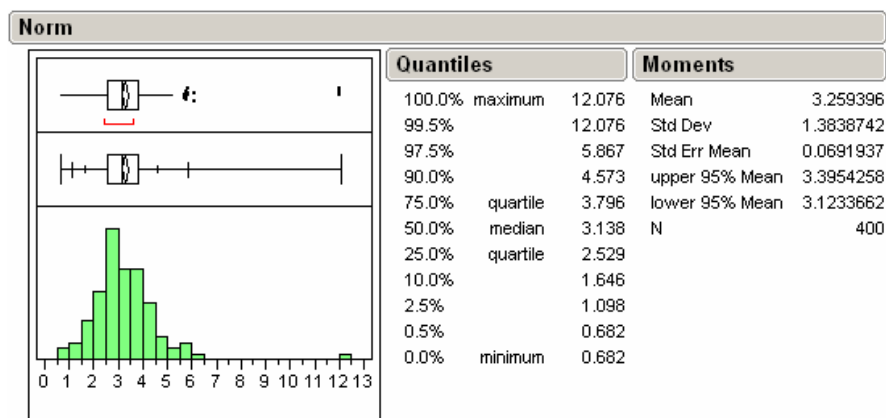
R&S Design 75957095



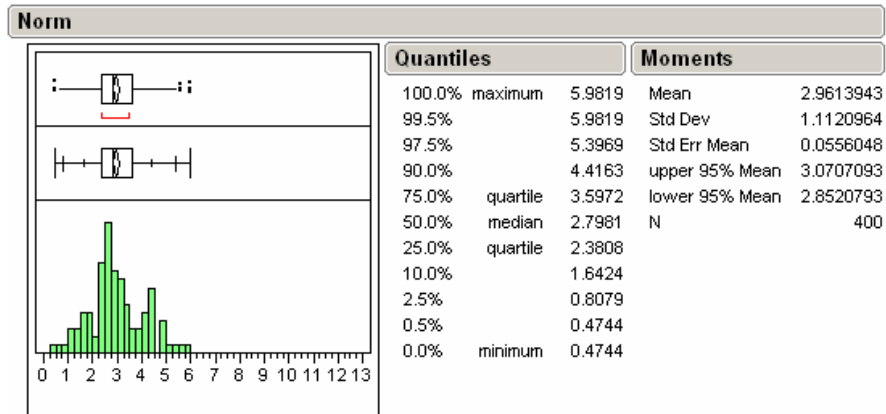
R&S Design 75958095



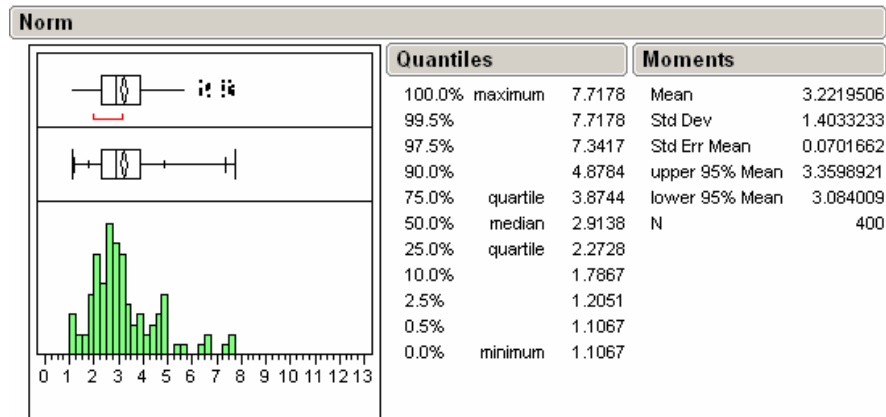
R&S Design 100907090



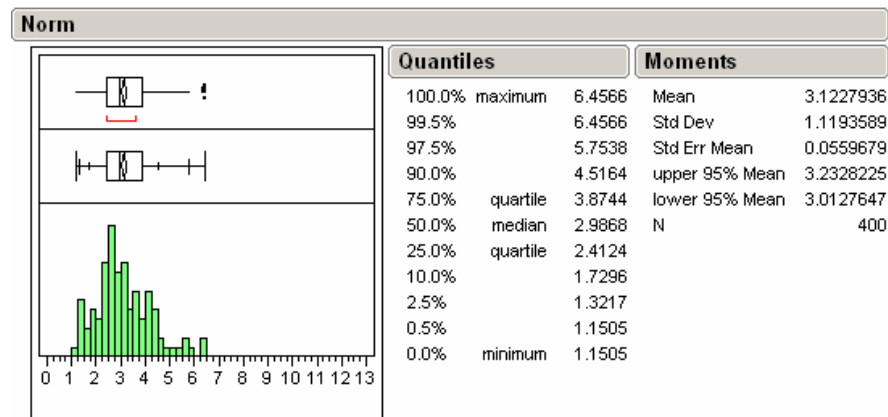
R&S Design 100908090



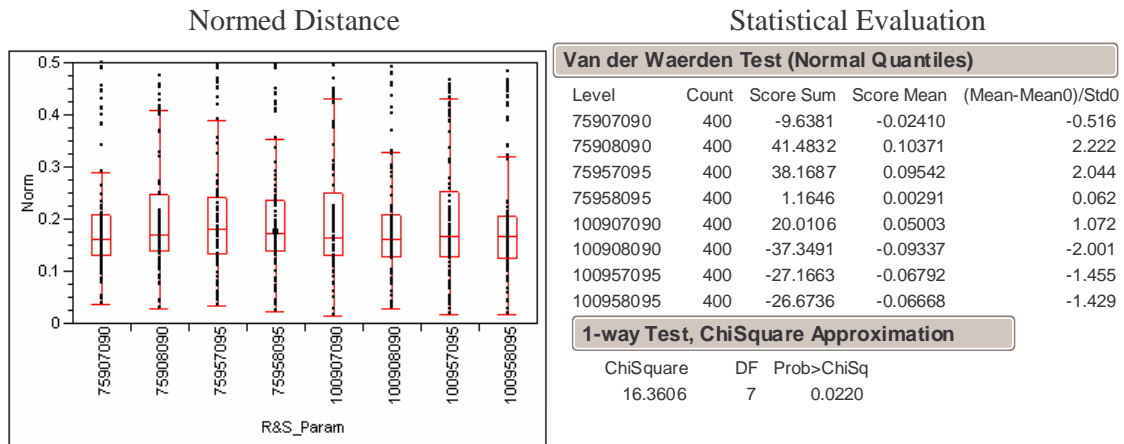
R&S Design 100957095



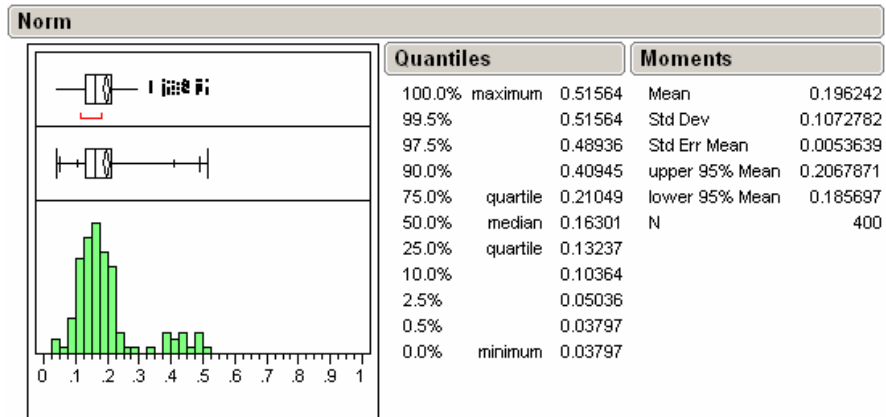
R&S Design 100958095



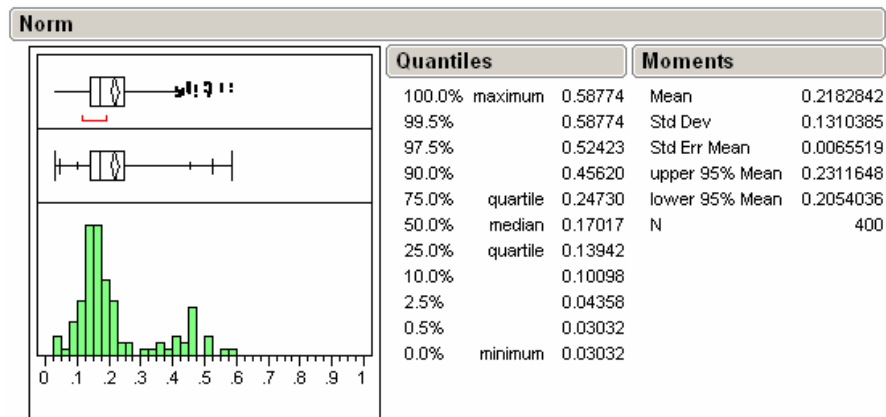
Test Problem 3



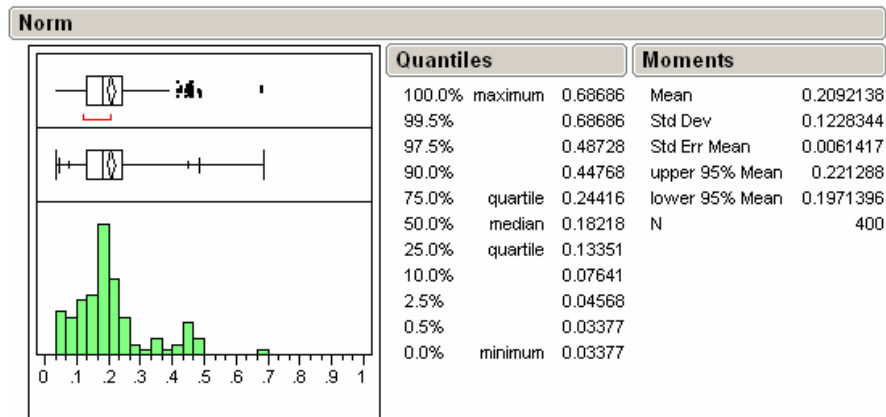
R&S Design 75907090



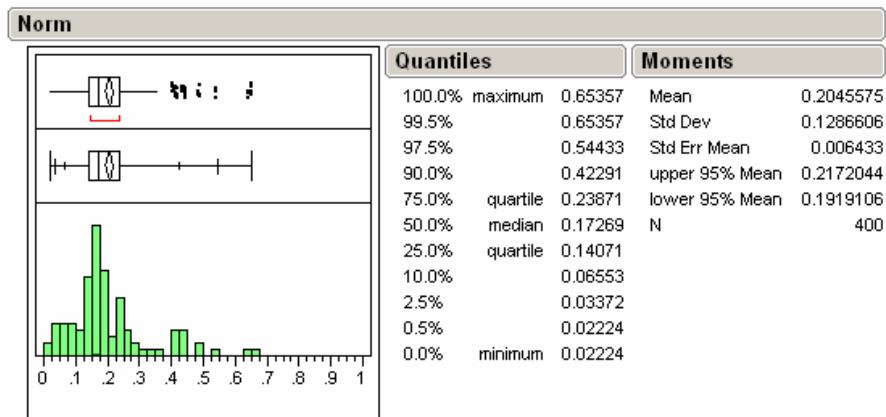
R&S Design 75908090



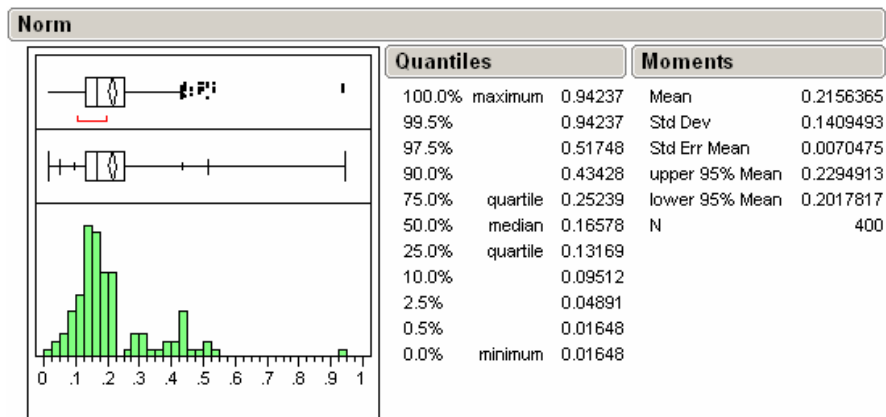
R&S Design 75957095



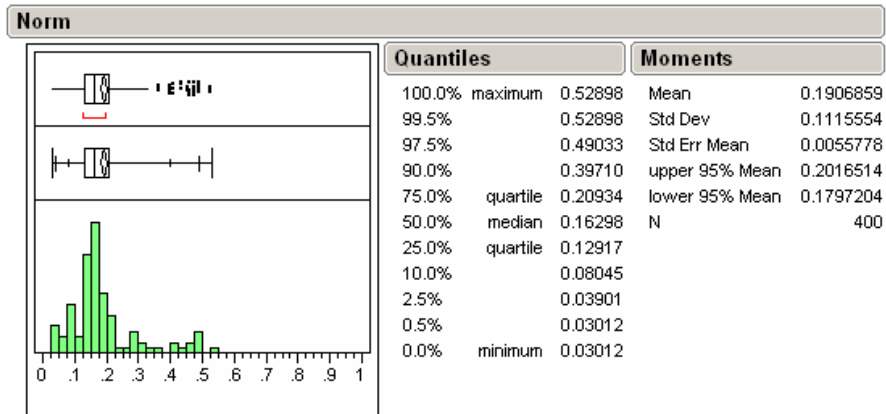
R&S Design 75958095



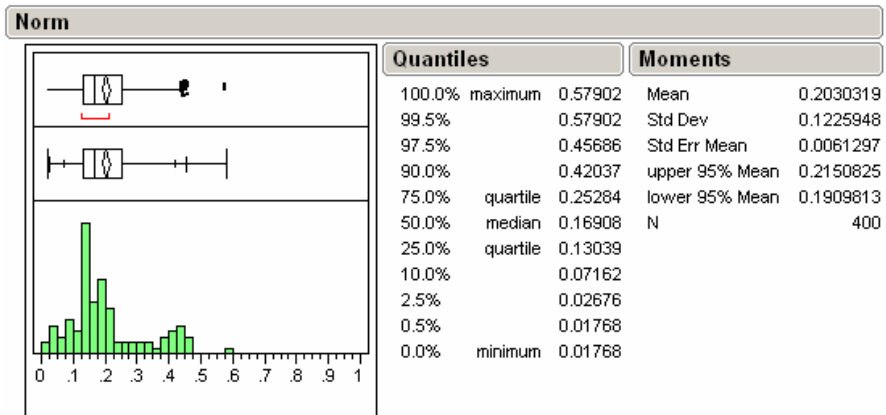
R&S Design 100907090



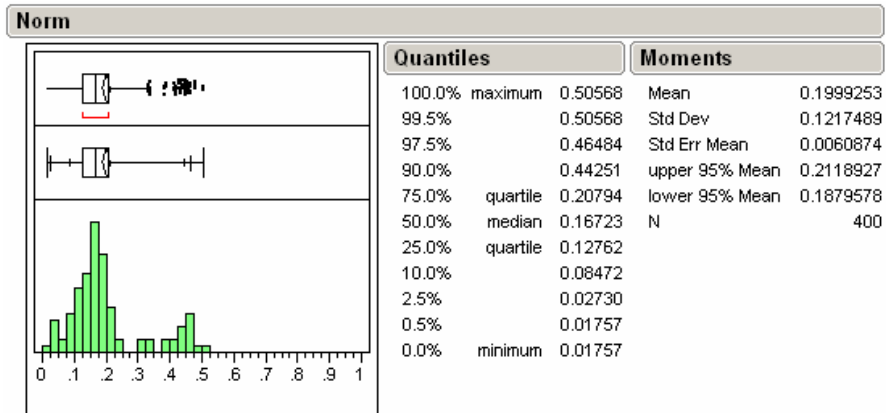
R&S Design 100908090



R&S Design 100957095



R&S Design 100958095

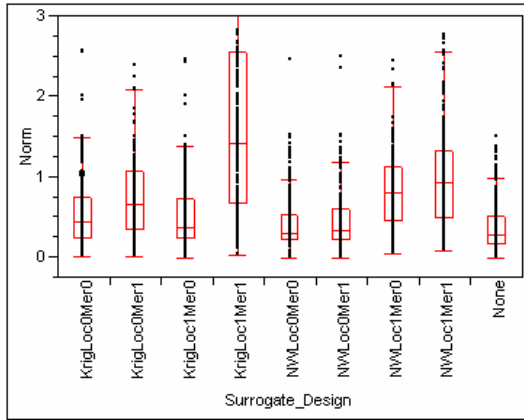


A.3 Main Run Data Summary

This appendix provides the details of the results of the main run used to determine if the use of surrogates improves the quality of the terminating solution. As noted in Section 4.4, the distance to the optimal point was used as the quality measure for each design setting. For each of the test problems, the box plot and statistical test are first presented and supported by the following individual design setting histograms and descriptive statistics. The labeling of each surrogate design is provided by the concatenation of the surrogate used with boolean indicators for the use of a local trust region and merit function, thus, KrigLoc1Mer0 represents the use of a Kriging surrogate with a local trust region but not a merit function.

Test Problem 1

Combined Norm

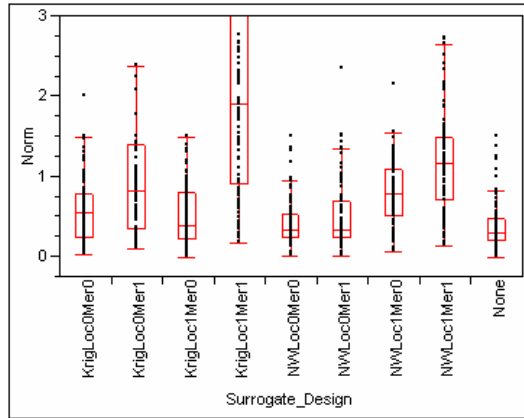


Statistical Evaluation

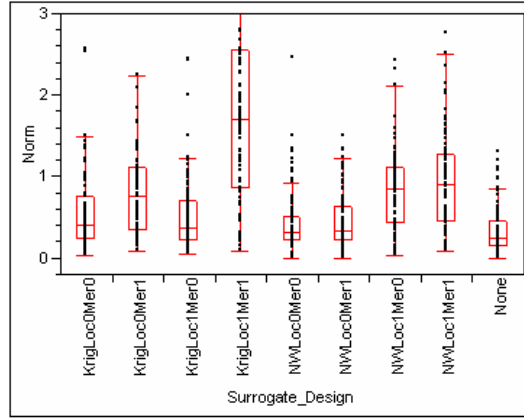
Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	0.141793	0.25027	0.453448	0.75915	1.05873
KrigLoc0Mer1	0.189633	0.35205	0.662344	1.073494	1.491015
KrigLoc1Mer0	0.126106	0.249604	0.38015	0.739609	1.144514
KrigLoc1Mer1	0.357085	0.674737	1.418628	2.564829	4.180494
NwLoc0Mer0	0.131703	0.23438	0.308773	0.530212	0.927676
NwLoc0Mer1	0.132827	0.226965	0.342431	0.614611	1.020827
NwLoc1Mer0	0.244269	0.462968	0.798893	1.137354	1.303267
NwLoc1Mer1	0.287717	0.499921	0.923669	1.328939	1.649144
None	0.100629	0.180662	0.290058	0.516337	0.817699

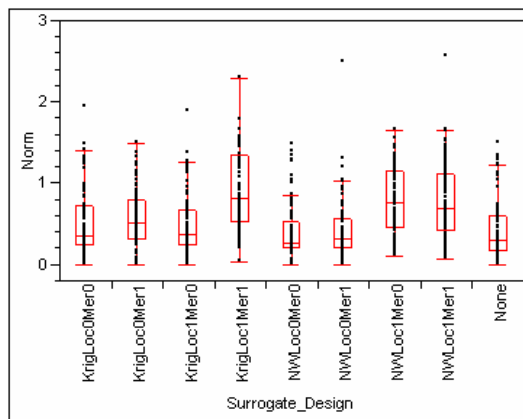
R&S Design 75907090



R&S Design 100908090



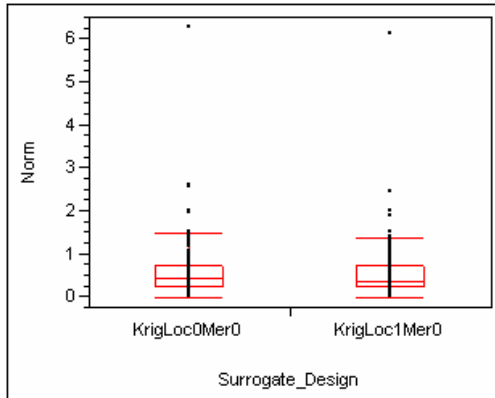
R&S Design 100958095



Test Problem 1

KrigLoc0Mer0 and KrigLoc1Mer0 Comparison

Normed Distance



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	92157.5	307.192	0.945
KrigLoc1Mer0	300	88142.5	293.808	-0.945

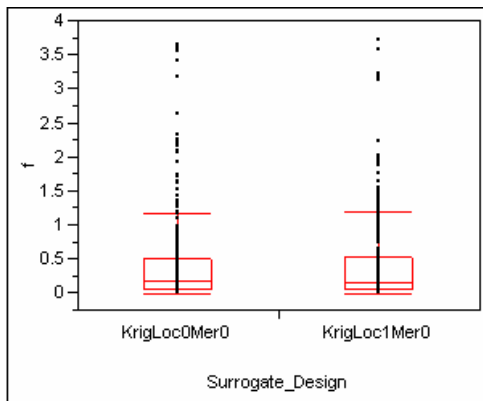
2-Sample Test, Normal Approximation

S	Z	Prob> Z
88142.5	-0.94532	0.3445

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.8941	1	0.3444

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	92877	309.590	1.284
KrigLoc1Mer0	300	87423	291.410	-1.284

2-Sample Test, Normal Approximation

S	Z	Prob> Z
87423	-1.28421	0.1991

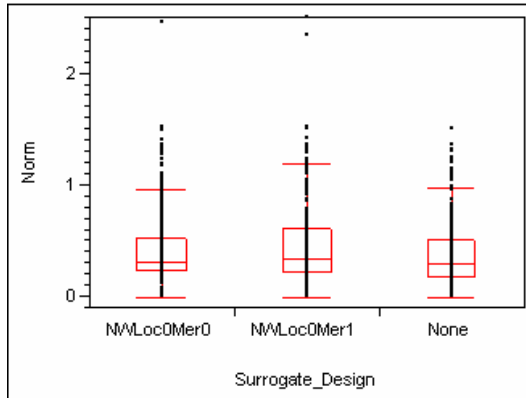
1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
1.6498	1	0.1990

Test Problem 1

NWLoc0Mer0, NWLoc0Mer1, and None Comparison

Normed Distance



Statistical Evaluation

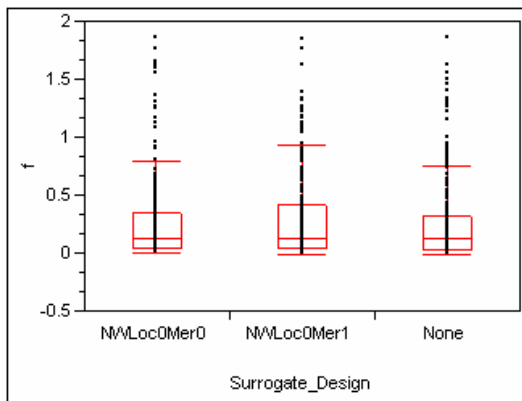
Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	136238.5	454.128	0.296
NWLoc0Mer1	300	141824.5	472.748	1.815
None	300	127387	424.623	-2.112

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
5.2287	2	0.0732

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

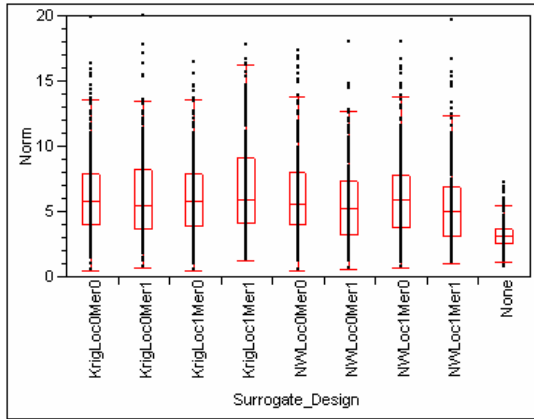
Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	135851.5	452.838	0.191
NWLoc0Mer1	300	138700.5	462.335	0.966
None	300	130898	436.327	-1.156

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
1.5379	2	0.4635

Test Problem 2

Combined Norm

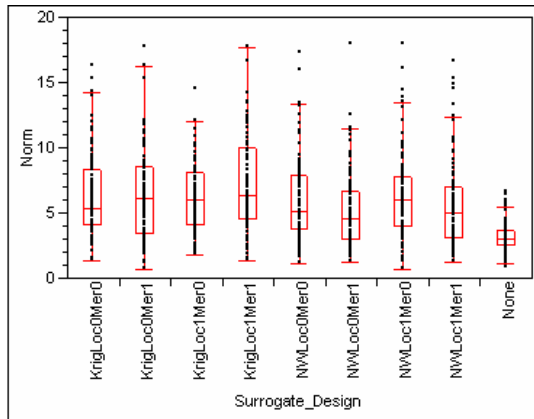


Statistical Evaluation

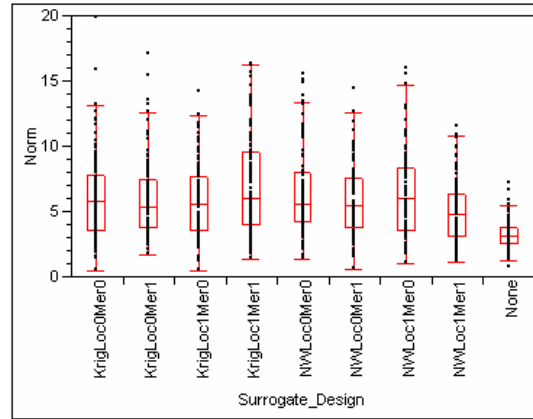
Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	2.873339	4.013382	5.779123	7.959836	10.52179
KrigLoc0Mer1	2.63661	3.689953	5.551306	8.279745	10.51901
KrigLoc1Mer0	2.805404	3.99156	5.802168	7.982192	9.77672
KrigLoc1Mer1	2.466074	4.214961	5.947904	9.121276	12.12097
NWLoc0Mer0	2.668076	4.083172	5.555668	8.00502	11.52101
NWLoc0Mer1	2.260656	3.273001	5.27244	7.421511	9.516558
NWLoc1Mer0	2.605152	3.883431	5.944353	7.886601	10.42115
NWLoc1Mer1	2.057595	3.151391	5.018045	6.970823	9.597397
None	2.196076	2.605398	3.124865	3.764776	4.357175

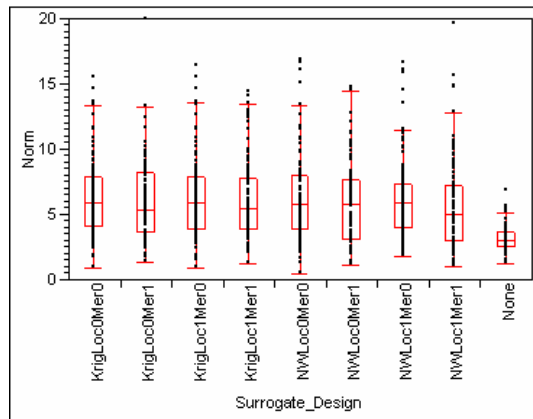
R&S Design 75907090



R&S Design 100908090



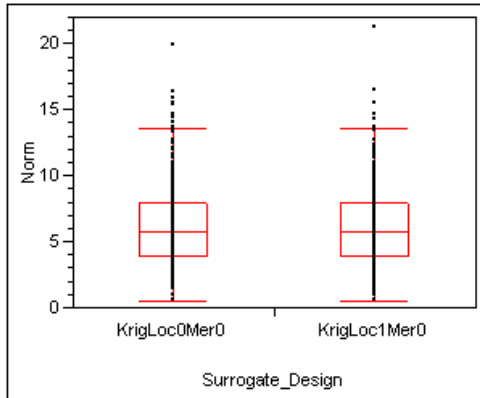
R&S Design 100958095



Test Problem 2

KrigLoc0Mer0 and KrigLoc1Mer0 Comparison

Normed Distance



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	90609	302.030	0.216
KrigLoc1Mer0	300	89691	298.970	-0.216

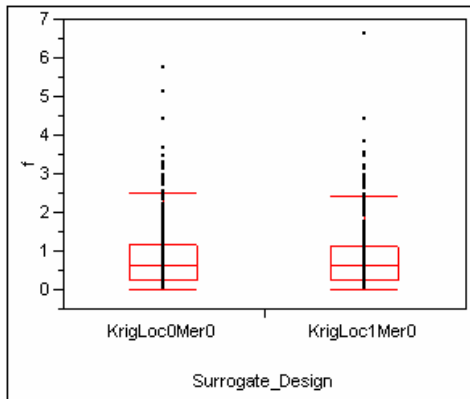
2-Sample Test, Normal Approximation

S	Z	Prob> Z
89691	-0.21596	0.8290

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.0467	1	0.8288

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	90776	302.587	0.295
KrigLoc1Mer0	300	89524	298.413	-0.295

2-Sample Test, Normal Approximation

S	Z	Prob> Z
89524	-0.29462	0.7683

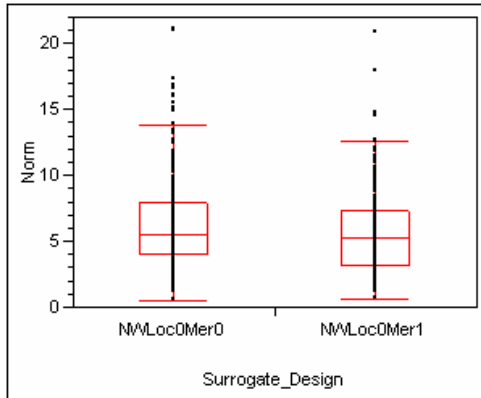
1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.0869	1	0.7681

Test Problem 2

NWLoc0Mer0 and NWLoc0Mer1 Comparison

Normed Distance



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	95154.5	317.182	2.357
NWLoc0Mer1	300	85145.5	283.818	-2.357

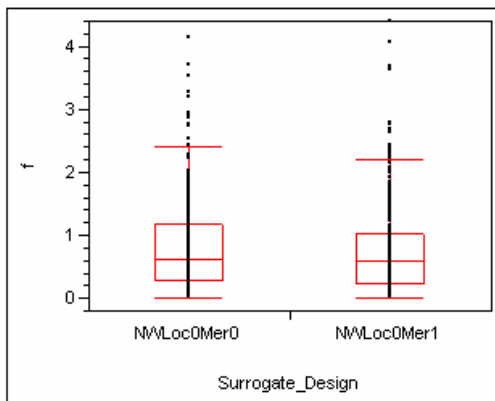
2-Sample Test, Normal Approximation

S	Z	Prob> Z
85145.5	-2.35695	0.0184

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
5.5563	1	0.0184

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	92768.5	309.228	1.233
NWLoc0Mer1	300	87531.5	291.772	-1.233

2-Sample Test, Normal Approximation

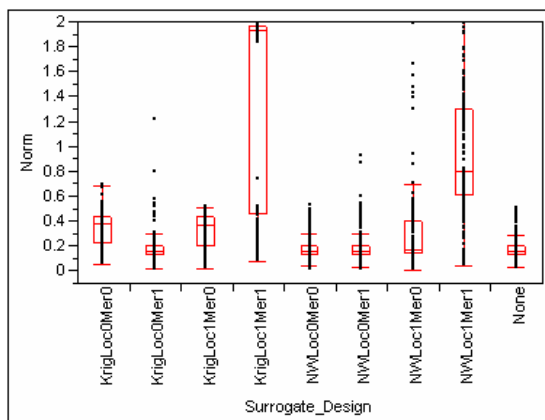
S	Z	Prob> Z
87531.5	-1.23311	0.2175

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
1.5211	1	0.2174

Test Problem 3

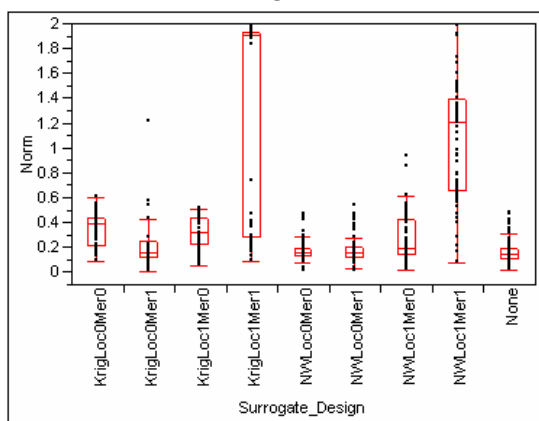
Combined Norm



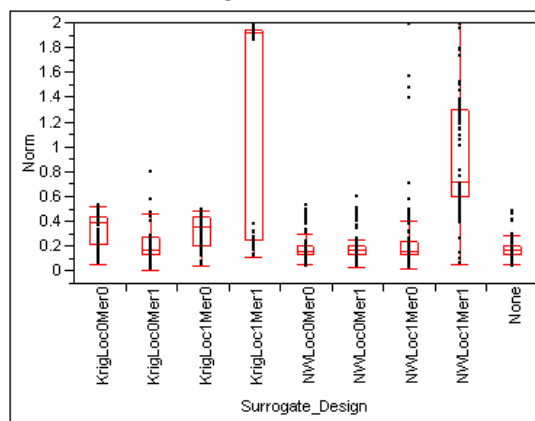
Statistical Evaluation

Quantiles					
Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	0.151032	0.231649	0.388473	0.439145	0.460418
KrigLoc0Mer1	0.08946	0.134483	0.164621	0.209701	0.282843
KrigLoc1Mer0	0.143989	0.209848	0.377724	0.441052	0.457555
KrigLoc1Mer1	0.209987	0.468505	1.936635	1.967288	1.990095
NWLoc0Mer0	0.099442	0.140174	0.16351	0.205942	0.346286
NWLoc0Mer1	0.100061	0.139356	0.167373	0.211763	0.415791
NWLoc1Mer0	0.095384	0.14602	0.178425	0.413017	0.483606
NWLoc1Mer1	0.453216	0.617499	0.799459	1.309955	1.99575
None	0.098791	0.13674	0.168278	0.20796	0.322285

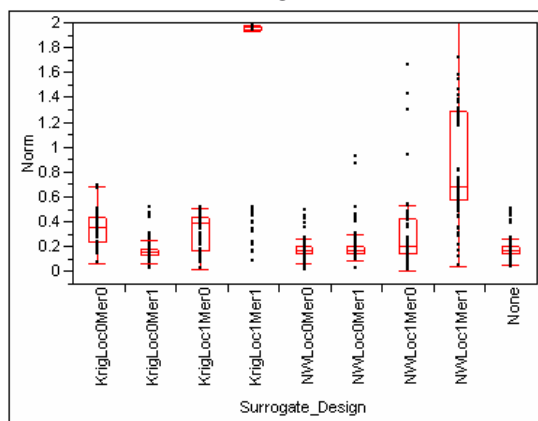
R&S Design 75907090



R&S Design 100908090



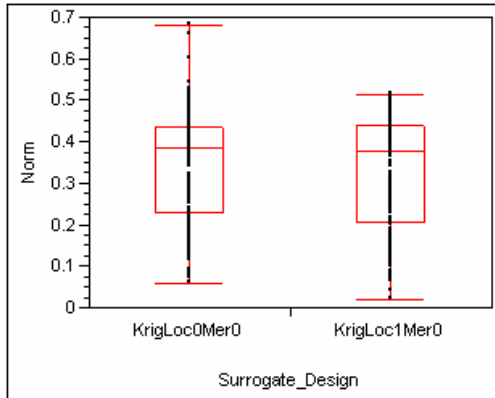
R&S Design 100958095



Test Problem 3

KrigLoc0Mer0 and KrigLoc1Mer0 Comparison

Normed Distance



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	91710.5	305.702	0.735
KrigLoc1Mer0	300	88589.5	295.298	-0.735

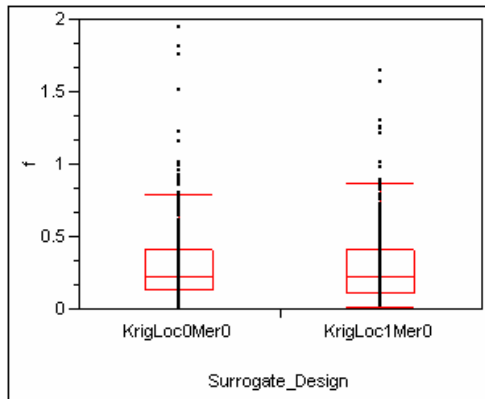
2-Sample Test, Normal Approximation

S	Z	Prob> Z
88589.5	-0.73478	0.4625

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.5402	1	0.4623

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
KrigLoc0Mer0	300	91123.5	303.745	0.458
KrigLoc1Mer0	300	89176.5	297.255	-0.458

2-Sample Test, Normal Approximation

S	Z	Prob> Z
89176.5	-0.45829	0.6467

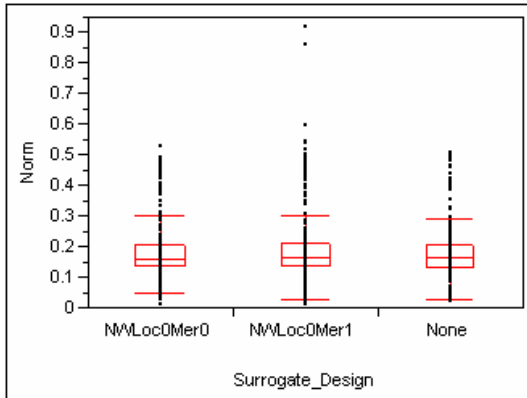
1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.2103	1	0.6466

Test Problem 3

NWLoc0Mer0, NWLoc0Mer1 and None Comparison

Normed Distance



Statistical Evaluation

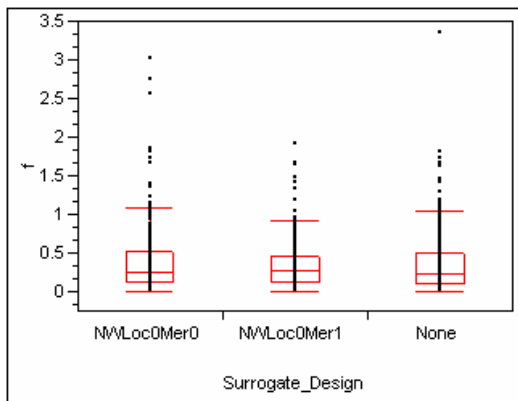
Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	134265	447.550	-0.241
NWLoc0Mer1	300	137605	458.683	0.668
None	300	133580	445.267	-0.427

1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.4575	2	0.7955

True Objective Function



Statistical Evaluation

Wilcoxon / Kruskal-Wallis Tests (Rank Sums)

Level	Count	Score Sum	Score Mean	(Mean-Mean0)/Std0
NWLoc0Mer0	300	137090	456.967	0.528
NWLoc0Mer1	300	136225	454.083	0.292
None	300	132135	440.450	-0.820

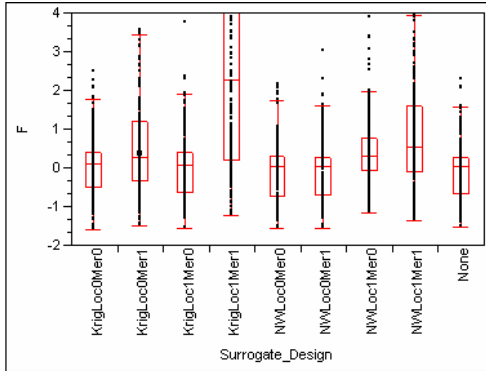
1-way Test, ChiSquare Approximation

ChiSquare	DF	Prob>ChiSq
0.6911	2	0.7078

A.4 R&S Procedure Adjustment Results

Test Problem 1

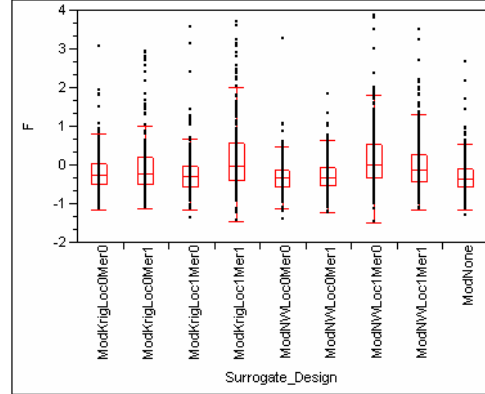
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	-0.92598	-0.48503	0.108955	0.422495	1.06597
KrigLoc0Mer1	-0.92956	-0.3034	0.296775	1.209625	6.00277
KrigLoc1Mer0	-0.96011	-0.60645	0.075999	0.42025	1.16797
KrigLoc1Mer1	-0.22095	0.203327	2.2827	21.342	61.1114
NWLoc0Mer0	-1.03187	-0.70592	0.039461	0.305915	0.650844
NWLoc0Mer1	-0.9934	-0.69384	0.051803	0.29853	0.754418
NWLoc1Mer0	-0.84534	-0.04441	0.309025	0.798075	1.38523
NWLoc1Mer1	-0.84282	-0.07153	0.54272	1.6254	3.8815
None	-1.03203	-0.63705	0.056497	0.271562	0.5642

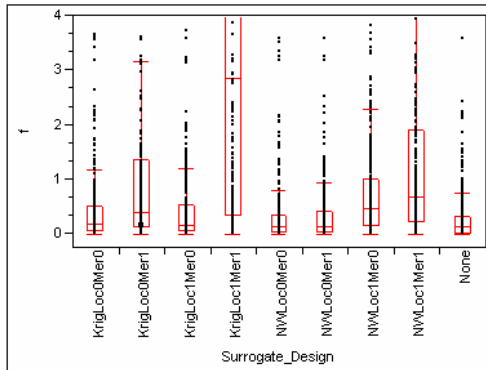
Modified R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	-0.71699	-0.48209	-0.26268	0.05637	0.441054
ModKrigLoc0Mer1	-0.72035	-0.48262	-0.20144	0.212612	2.54073
ModKrigLoc1Mer0	-0.76898	-0.54724	-0.28862	-0.02475	0.223169
ModKrigLoc1Mer1	-0.61419	-0.38221	-0.01834	0.585703	2.54866
ModNWLoc0Mer0	-0.74421	-0.53765	-0.31666	-0.12487	0.157835
ModNWLoc0Mer1	-0.70982	-0.52729	-0.31498	-0.05775	0.230275
ModNWLoc1Mer0	-0.59256	-0.30514	0.004371	0.54728	1.27644
ModNWLoc1Mer1	-0.62639	-0.40874	-0.10498	0.283445	0.963726
ModNone	-0.7617	-0.55659	-0.33704	-0.09477	0.213755

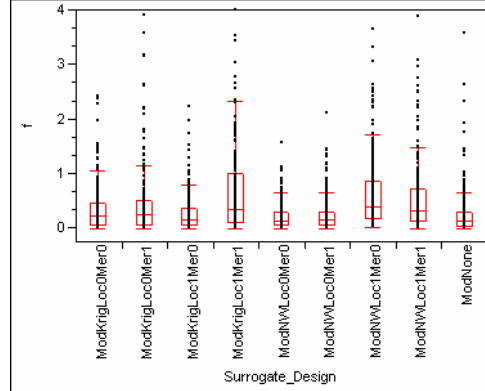
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	0.018823	0.0795	0.190926	0.526644	1.316783
KrigLoc0Mer1	0.045361	0.130981	0.401911	1.372592	26.00287
KrigLoc1Mer0	0.010442	0.066388	0.160579	0.532767	1.377032
KrigLoc1Mer1	0.104037	0.349397	2.862748	357.4401	604.1474
NWLoc0Mer0	0.011117	0.047069	0.136719	0.349507	0.80145
NWLoc0Mer1	0.009573	0.052716	0.134433	0.419762	1.070177
NWLoc1Mer0	0.071668	0.168019	0.47774	1.019575	2.076141
NWLoc1Mer1	0.073508	0.224128	0.690541	1.903625	6.811658
None	0.007281	0.032205	0.137564	0.332082	0.855352

Modified R&S Procedure

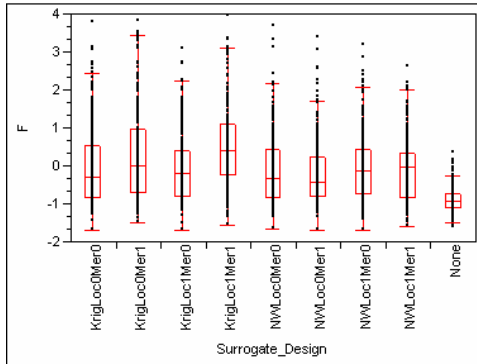


Quantiles

Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	0.030646	0.079851	0.231983	0.473079	0.758084
ModKrigLoc0Mer1	0.025844	0.07554	0.247169	0.514493	3.131908
ModKrigLoc1Mer0	0.024298	0.071111	0.165326	0.372709	0.731697
ModKrigLoc1Mer1	0.051976	0.112076	0.340956	1.024658	4.294366
ModNWLoc0Mer0	0.02277	0.063947	0.147813	0.299645	0.500168
ModNWLoc0Mer1	0.018738	0.067017	0.15843	0.313768	0.607162
ModNWLoc1Mer0	0.079722	0.176804	0.395631	0.864855	1.714046
ModNWLoc1Mer1	0.054815	0.146886	0.319635	0.721275	1.344146
ModNone	0.017621	0.048951	0.147823	0.298284	0.559412

Test Problem 2

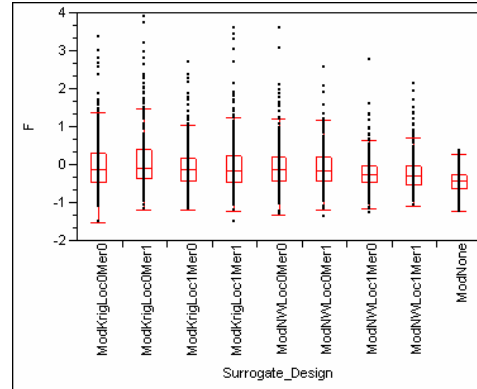
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	-0.99971	-0.81612	-0.28567	0.535453	1.27745
KrigLoc0Mer1	-0.98078	-0.69523	0.027073	0.97654	3.04455
KrigLoc1Mer0	-1.02275	-0.77672	-0.18204	0.432145	1.27745
KrigLoc1Mer1	-0.74777	-0.22311	0.407095	1.128875	2.79799
NWLoc0Mer0	-1.04465	-0.82505	-0.31495	0.438325	1.18745
NWLoc0Mer1	-1.01137	-0.77523	-0.41217	0.260085	0.907198
NWLoc1Mer0	-0.98489	-0.73111	-0.11371	0.434955	1.06983
NWLoc1Mer1	-1.04346	-0.80376	-0.03194	0.347608	0.789408
None	-1.21618	-1.07108	-0.90685	-0.72816	-0.50436

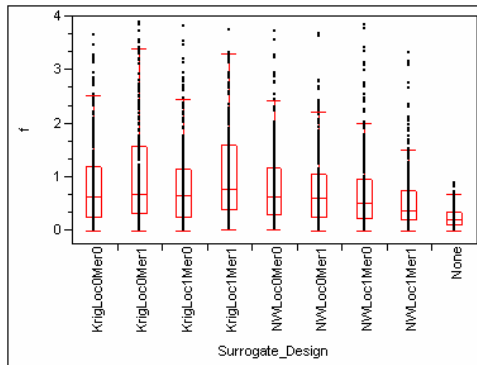
Modified R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	-0.62332	-0.43376	-0.12279	0.30105	0.923555
ModKrigLoc0Mer1	-0.63772	-0.35651	-0.07594	0.419698	1.60339
ModKrigLoc1Mer0	-0.61278	-0.40637	-0.11949	0.196968	0.647549
ModKrigLoc1Mer1	-0.7032	-0.43528	-0.16037	0.238968	0.926588
ModNWLoc0Mer0	-0.64132	-0.42817	-0.12803	0.231753	0.691485
ModNWLoc0Mer1	-0.63451	-0.40867	-0.15459	0.228718	0.579323
ModNWLoc1Mer0	-0.71671	-0.46036	-0.24761	-0.00233	0.223772
ModNWLoc1Mer1	-0.71786	-0.50881	-0.26754	-0.0155	0.391129
ModNone	-0.81529	-0.62446	-0.42919	-0.23596	-0.05539

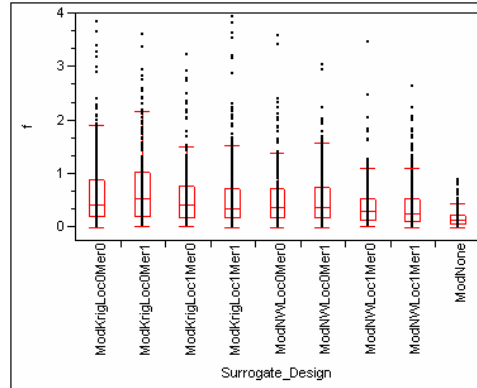
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	0.121571	0.267207	0.641923	1.19316	2.082607
KrigLoc0Mer1	0.168234	0.335471	0.692988	1.573814	3.395072
KrigLoc1Mer0	0.11768	0.252543	0.655294	1.145067	1.739752
KrigLoc1Mer1	0.179431	0.400148	0.786821	1.611666	3.135825
NWLoc0Mer0	0.153532	0.298775	0.637065	1.181302	1.859797
NWLoc0Mer1	0.142914	0.259851	0.609259	1.050145	1.688589
NWLoc1Mer0	0.113322	0.243978	0.526066	0.96028	1.697488
NWLoc1Mer1	0.067966	0.21262	0.38459	0.742093	1.393495
None	0.053698	0.116962	0.204723	0.344811	0.539051

Modified R&S Procedure

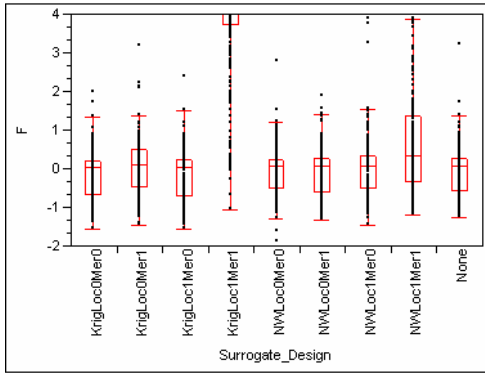


Quantiles

Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	0.086734	0.202793	0.433352	0.885896	1.629481
ModKrigLoc0Mer1	0.083585	0.218244	0.551076	1.036486	1.942927
ModKrigLoc1Mer0	0.088778	0.195346	0.42009	0.767944	1.258362
ModKrigLoc1Mer1	0.083698	0.176274	0.348271	0.723024	1.455409
ModNWLoc0Mer0	0.091718	0.176525	0.383691	0.719563	1.185811
ModNWLoc0Mer1	0.09876	0.186555	0.371616	0.752262	1.25283
ModNWLoc1Mer0	0.053335	0.138543	0.311351	0.542609	0.907939
ModNWLoc1Mer1	0.057252	0.123268	0.252135	0.531049	0.969244
ModNone	0.046082	0.075756	0.133034	0.227792	0.367479

Test Problem 3

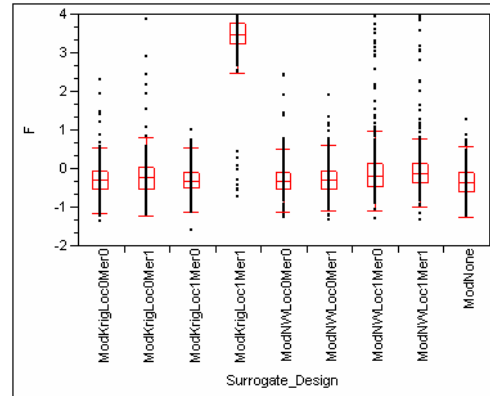
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	-0.98024	-0.63477	0.063467	0.206845	0.474602
KrigLoc0Mer1	-0.93332	-0.443	0.1088	0.514598	28.513
KrigLoc1Mer0	-0.96404	-0.67035	0.062208	0.263853	0.522427
KrigLoc1Mer1	1.20548	3.736325	4.7131	7.03615	10.3296
NWLoc0Mer0	-0.88065	-0.48029	0.08402	0.256855	0.569916
NWLoc0Mer1	-0.93165	-0.58655	0.074842	0.2668	0.510292
NWLoc1Mer0	-0.86524	-0.47518	0.095374	0.347477	0.721095
NWLoc1Mer1	-0.7328	-0.31747	0.362165	1.3743	3.82338
None	-0.90168	-0.55789	0.075631	0.290478	0.540055

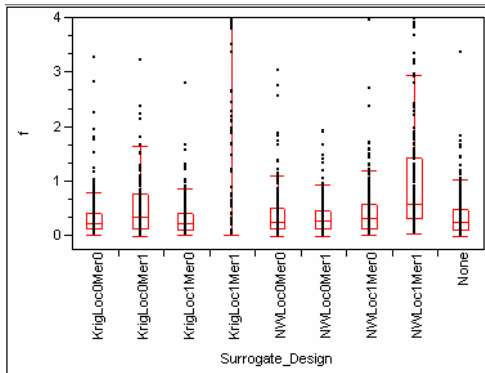
Modified R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	-0.74184	-0.52022	-0.2917	-0.06141	0.174166
ModKrigLoc0Mer1	-0.73613	-0.50058	-0.23317	0.048396	2.84933
ModKrigLoc1Mer0	-0.67427	-0.49466	-0.3006	-0.07138	0.167302
ModKrigLoc1Mer1	2.9276	3.24875	3.48945	3.7756	4.06281
ModNWLoc0Mer0	-0.72207	-0.50959	-0.31232	-0.09491	0.205445
ModNWLoc0Mer1	-0.73923	-0.50484	-0.27814	-0.05958	0.203608
ModNWLoc1Mer0	-0.62331	-0.43377	-0.19472	0.135865	0.873579
ModNWLoc1Mer1	-0.55912	-0.34954	-0.1136	0.139703	0.733484
ModNone	-0.81668	-0.57386	-0.35295	-0.09683	0.136145

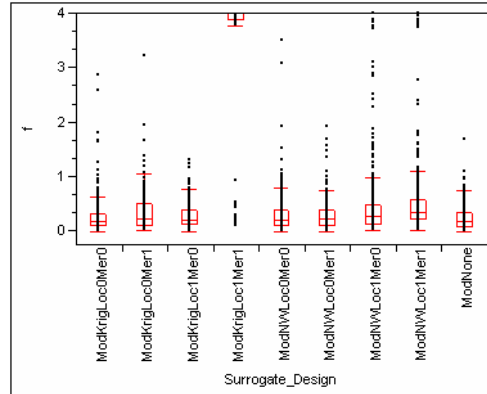
Original R&S Procedure



Quantiles

Level	10%	25%	Median	75%	90%
KrigLoc0Mer0	0.083977	0.136903	0.224262	0.412913	0.713729
KrigLoc0Mer1	0.060742	0.135336	0.346403	0.772988	28.51369
KrigLoc1Mer0	0.065662	0.11535	0.233313	0.42132	0.660879
KrigLoc1Mer1	1.453449	4.023265	4.914021	7.322987	10.35761
NWLoc0Mer0	0.064632	0.138931	0.265122	0.525214	0.814552
NWLoc0Mer1	0.059321	0.137515	0.276099	0.462575	0.746003
NWLoc1Mer0	0.069852	0.145779	0.321323	0.589814	0.940543
NWLoc1Mer1	0.190882	0.319809	0.598734	1.437462	3.995306
None	0.051898	0.118992	0.249486	0.501742	0.847007

Modified R&S Procedure

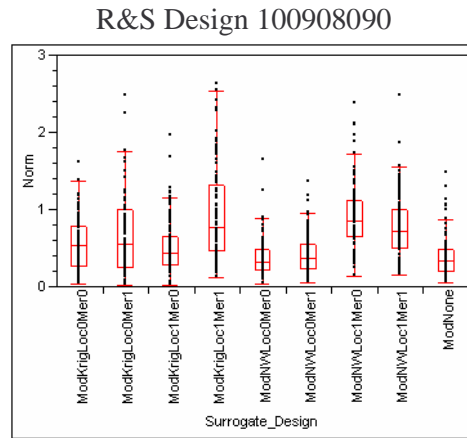
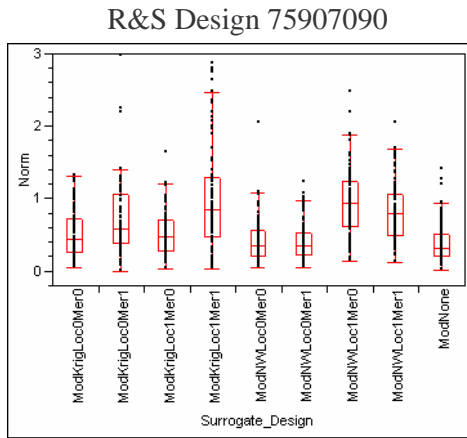
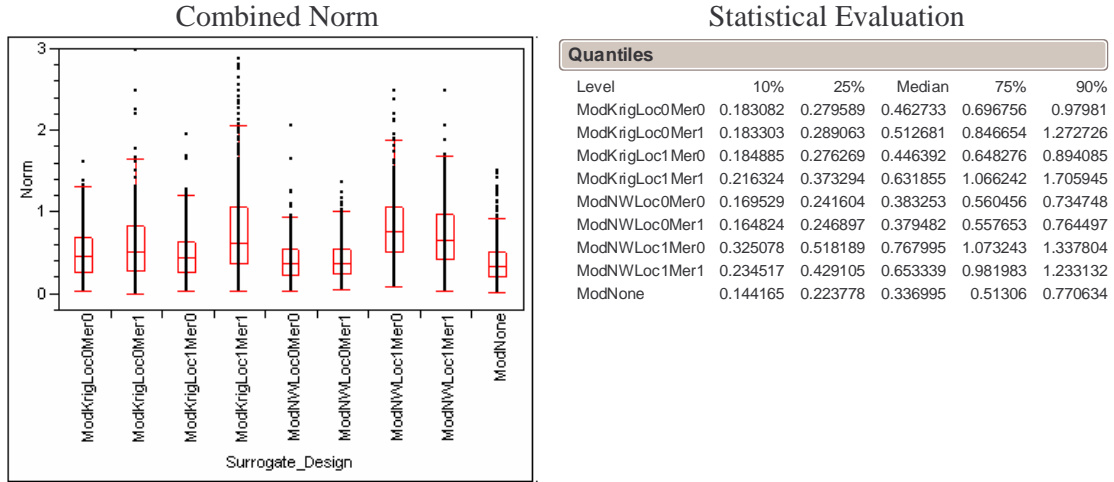


Quantiles

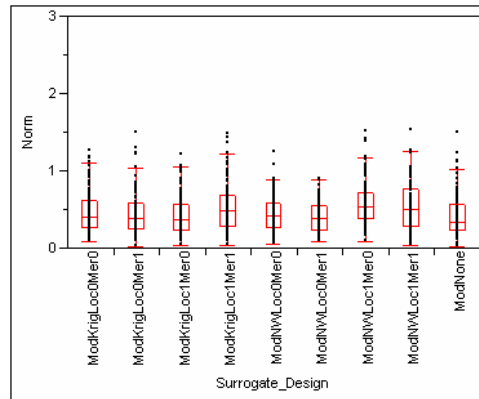
Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	0.075942	0.116269	0.181236	0.322895	0.546881
ModKrigLoc0Mer1	0.06924	0.115305	0.238733	0.506078	1.943926
ModKrigLoc1Mer0	0.077771	0.129834	0.208359	0.39016	0.581663
ModKrigLoc1Mer1	3.840532	3.894317	4.012275	4.178733	4.407357
ModNWLoc0Mer0	0.055592	0.111516	0.217124	0.389035	0.601266
ModNWLoc0Mer1	0.049932	0.109708	0.229312	0.401238	0.606474
ModNWLoc1Mer0	0.079445	0.133717	0.284633	0.491141	1.396926
ModNWLoc1Mer1	0.173232	0.238003	0.36267	0.586544	1.377932
ModNone	0.043354	0.086942	0.187836	0.356152	0.507798

A.5 Modified Main Run Data Summary

Test Problem 1

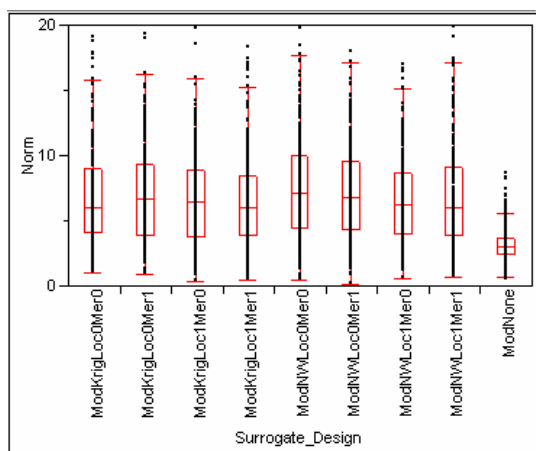


R&S Design 100958095



Test Problem 2

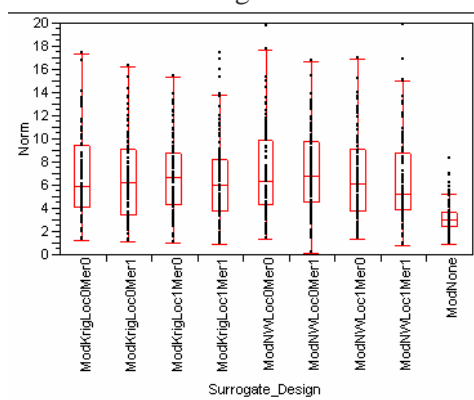
Combined Norm



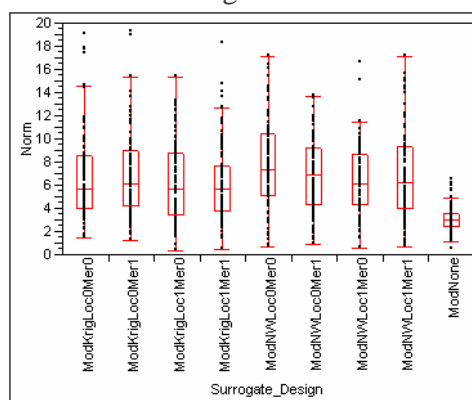
Statistical Evaluation

Quantiles					
Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	2.81051	4.113336	6.068051	9.030545	11.43659
ModKrigLoc0Mer1	2.585913	3.983391	6.76084	9.375185	12.49002
ModKrigLoc1Mer0	2.712457	3.865991	6.551751	8.911808	11.53997
ModKrigLoc1Mer1	2.411108	3.936627	6.042433	8.523032	11.51943
ModNWLoc0Mer0	2.705639	4.508757	7.173094	10.00994	13.44253
ModNWLoc0Mer1	2.811929	4.430229	6.852268	9.626155	12.01134
ModNWLoc1Mer0	2.457249	4.099972	6.249081	8.716219	10.79959
ModNWLoc1Mer1	2.406234	3.893905	6.000206	9.219319	12.11883
ModNone	1.768445	2.523795	3.093501	3.766776	4.763328

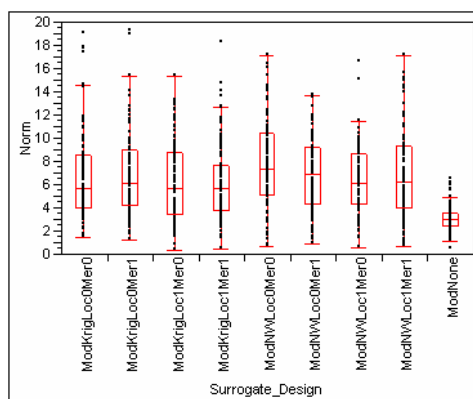
R&S Design 75907090



R&S Design 100908090

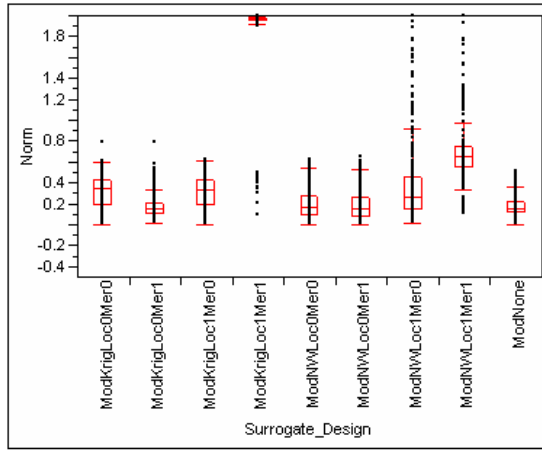


R&S Design 100958095



Test Problem 3

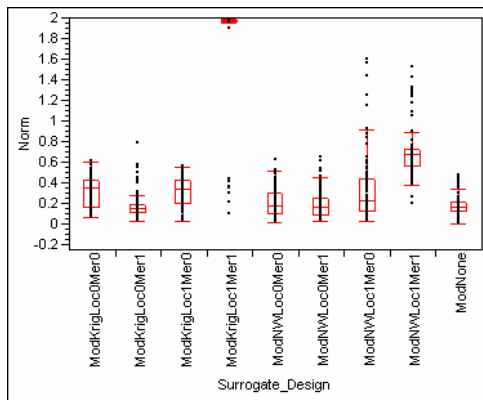
Combined Norm



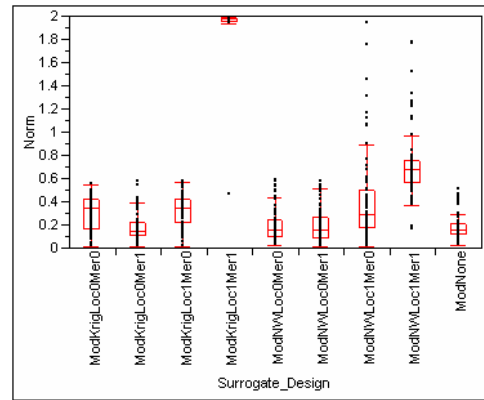
Statistical Evaluation

Quantiles					
Level	10%	25%	Median	75%	90%
ModKrigLoc0Mer0	0.131378	0.199246	0.355268	0.433448	0.474734
ModKrigLoc0Mer1	0.077898	0.120507	0.154298	0.216067	0.37532
ModKrigLoc1Mer0	0.138444	0.198441	0.346302	0.43218	0.474837
ModKrigLoc1Mer1	1.951534	1.966005	1.981919	1.995479	2.006684
ModNWLoc0Mer0	0.051673	0.09949	0.173924	0.287299	0.451325
ModNWLoc0Mer1	0.049727	0.0954	0.166327	0.275793	0.396389
ModNWLoc1Mer0	0.081499	0.154015	0.266646	0.464458	1.211394
ModNWLoc1Mer1	0.441	0.563356	0.662058	0.750048	1.274063
ModNone	0.0839	0.127854	0.165218	0.226187	0.395346

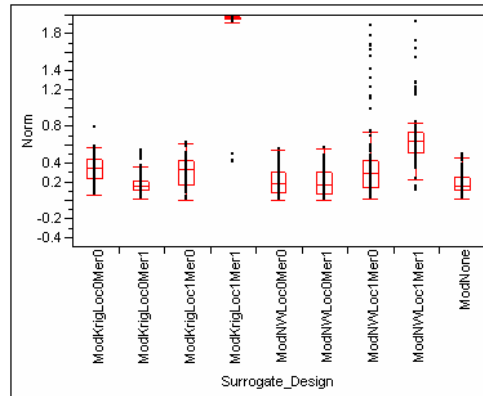
R&S Design 75907090



R&S Design 100908090



R&S Design 100958095



BIBLIOGRAPHY

- [1] ABRAMSON, M. A. Nonlinear optimization with mixed variables and derivatives-matlab (NOMADm). Software. Available from Mark Abramson's NOMADm Page, <http://en.ait.edu/ENC/Faculty/MAbramson/nomadm.html>.
- [2] ABRAMSON, M. A. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Rice University, Department of Computational and Applied Mathematics, 2002.
- [3] ABRAMSON, M. A. Second order behavior of pattern search algorithms. Tech. Rep. TR04-03, Rice University, Department of Computational and Applied Mathematics, Rice University, Houston TX, 2004.
- [4] ABRAMSON, M. A., AUDET, C., AND DENNIS, JR., J. E. Generalized pattern search with derivative information. Tech. Rep. TR02-10, Department of Computational and Applied Mathematics, Rice University, Houston TX, 2002.
- [5] AUDET, C., AND DENNIS, JR., J. E. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization* 11, 3 (2000), 573–594.
- [6] AUDET, C., AND DENNIS, JR., J. E. Analysis of generalized pattern searches. *SIAM Journal on Optimization* 13, 3 (2003), 889–903.
- [7] AUDET, C., AND DENNIS, JR., J. E. Mesh adaptive direct search algorithms for constrained optimization. Tech. Rep. TR04-02, Department of Computational and Applied Mathematics, Rice University, Houston TX, 2004.
- [8] AUDET, C., AND DENNIS, JR., J. E. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization* 14, 4 (2004), 980–1010.
- [9] BOOKER, A., DENNIS, JR., J. E., FRANK, P., SERAFINI, D., AND TORCZON, V. Optimization using surrogate objectives on a helicopter test example. In *Optimal Design* (Philadelphia, 1998), J. Burns and E. Cliff, Eds., SIAM.
- [10] BOOKER, A. J., DENNIS, JR., J. E., FRANK, P. D., SERAFINI, D. B., TORCZON, V., AND TROSSET, M. W. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization* 17, 1 (1999), 1–13.
- [11] BOX, G. E. P. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics* 6 (1957), 81–101.
- [12] BREZHNEVA, O. A., AND DENNIS, JR., J. E. Pattern search methods for linearly constrained minimization in the presence of degeneracy. Tech. Rep. TR03-09, Rice University, Department of Computational and Applied Mathematics, Rice University, Houston TX, 2003.
- [13] CONN, A. R., GOULD, N. I. M., AND TOINT, P. L. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and

- simple bound. *SIAM Journal on Numerical Analysis* 28, 2 (1991), 545–572.
- [14] COOPE, I. D., AND PRICE, C. J. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization* 11, 4 (2001), 859–869.
 - [15] CURRIN, C., MITCHELL, T., MORRIS, M., AND YLVISAKER, D. A Bayesian approach to the design and analysis of computer experiments. Tech. Rep. 6498, Oak Ridge National Laboratories, 1988.
 - [16] CUSTÓDIO, A. L., AND VICENTE, L. N. Using sampling and simplex derivatives in pattern search methods. Tech. Rep. Preprint 04-35, Department of Mathematics, University of Coimbra, 2004.
 - [17] DAVIS, C. Theory of positive linear dependence. *American Journal of Mathematics* 76, 4 (1954), 733–746.
 - [18] DE JONG, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.
 - [19] DENNIS, JR., J. E., AND SCHNABEL, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., Englewood Cliffs NJ, 1983.
 - [20] DENNIS, JR., J. E., AND TORCZON, V. Direct search methods on parallel machines. *SIAM Journal on Optimization* 1, 4 (1991), 448–474.
 - [21] DOLAN, E. D., LEWIS, R. M., AND TORCZON, V. On the local convergence of pattern search. *SIAM Journal on Optimization* 14, 2 (2003), 567–583.
 - [22] FABIAN, V. Note on Anderson’s sequential procedures with triangular boundary. *Annals of Statistics* 2 (1974), 170–176.
 - [23] FLETCHER, R., AND LEYFFER, S. Nonlinear programming without a penalty function. *Mathematical Programming* 91 (2002), 239–269.
 - [24] HÄRDLE, W. *Applied Nonparametric Regression*. Cambridge University Press, NY, 1990.
 - [25] HARTMANN, M. An improvement on Paulson’s sequential ranking procedure. *Sequential Analysis* 7 (1988), 363–372.
 - [26] HOLLAND, J. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor MI, 1975.
 - [27] HONG, L. J., AND NELSON, B. L. An indifference-zone selection procedure with minimum switching and sequential sampling. In *Proceedings of the 2003 Winter Simulation Conference* (Piscataway, New Jersey, 2003), S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, Eds., Institute of Electrical and Electronics Engineers, pp. 474–480.
 - [28] HOOKE, R., AND JEEVES, T. A. "Direct search" solution of numerical and

- statistical problems. *Journal of the Association of Computing Machinery* 8 (1961), 212–229.
- [29] KIM, S. H., AND NELSON, B. L. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation* 11, 3 (2001), 251–273.
 - [30] KIRKPATRICK, S., GELATT, JR., C., AND VECCHI, M. P. Optimization by simulated annealing. *Science* 220 (1983), 671–680.
 - [31] KOLDA, T. G., LEWIS, R. M., AND TORCZON, V. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review* 46 (2003), 385–482.
 - [32] KRIGE, D. G. A statistical approach to some mine problems and allied problems at the witwatersrand. Master’s thesis, University of Witwatersrand, 1951.
 - [33] LAW, A. M., AND KELTON, W. D. *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, NY, 2000.
 - [34] LEWIS, R. M., AND TORCZON, V. Rank ordering and positive bases in pattern search algorithms. Tech. Rep. ICASE 96-71, NASA Langley Research Center, 1996.
 - [35] LEWIS, R. M., AND TORCZON, V. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization* 9, 4 (1999), 1082–1099.
 - [36] LEWIS, R. M., AND TORCZON, V. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization* 10, 3 (2000), 917–941.
 - [37] LEWIS, R. M., AND TORCZON, V. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization* 12, 4 (2002), 1075–1089.
 - [38] LUCIDI, S., PICCIAILLI, V., AND SCIANDRONE, M. An algorithm model for mixed variable programming. Tech. Rep. 17-02, Department of Computer and Systems Science “Antonio Ruberti”, University of Rome, 2002.
 - [39] LUCIDI, S., AND SCIANDRONE, M. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization* 13, 1 (2002), 97–116.
 - [40] MCKAY, M. D., BECKMAN, R. J., AND CONOVER, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21, 2 (1979), 239–245.
 - [41] METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A., AND TELLER, E. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21 (1953), 1087–1092.
 - [42] NADARAYA, E. A. On estimating regression. *Theory of Probability and Its*

Applications 9 (1964), 141–142.

- [43] NASH, S., AND SOFER, A. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.
- [44] NELDER, J. A., AND MEAD, R. A simplex method for function minimization. *The Computer Journal* 7, 4 (1965), 308–313.
- [45] NELSON, B. L., SWANN, J., GOLDSMAN, D., AND SONG, W. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research* 49, 6 (2001), 950–963.
- [46] NOCEDAL, J., AND WRIGHT, S. *Numerical Optimization*. Springer, NY, 1999.
- [47] PAULSON, E. A sequential procedure for selecting the population with the largest mean from k normal populations. *Annals of Mathematical Statistics* 35 (1964), 174–180.
- [48] PICHITLAMKEN, J., AND NELSON, B. L. Selection-of-the-best procedures for optimization via simulation. In *Proceedings of the 2001 Winter Simulation Conference* (Piscataway NJ, 2001), B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, Eds., Institute of Electrical and Electronics Engineers, pp. 401–407.
- [49] PICHITLAMKEN, J., AND NELSON, B. L. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 13, 2 (2003), 155–179.
- [50] RECHENBERG, I. *Evolutionstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.
- [51] RINOTT, Y. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics* A7, 8 (1978), 799–811.
- [52] SCHITTKOWSKI, K. *More Test Examples for Nonlinear Programming Codes*. Springer-Verlag, Berlin, Heidelberg, NY, 1987. Lecture Notes in Economics and Mathematical Systems No. 282.
- [53] SHESKIN, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures*, 2nd ed. Chapman & Hall / CRC, Boca Raton FL, 2000.
- [54] SPALL, J. C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley and Sons, Hoboken NJ, 2003.
- [55] SRIVER, T. A. *Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems*. PhD thesis, Air Force Institute of Technology, 2004.
- [56] SRIVER, T. A., AND CHRISSIS, J. W. Combined pattern search and ranking and selection for simulation optimization. In *Proceedings of the 2004 Winter Simulation Conference* (Piscataway NJ, 2004), R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, Eds., Institute of Electrical and Electronics Engineers, pp. 645–653.

- [57] TONG, Y. L. *Probability Inequalities in Multivariate Distributions*. Academic Press, NY, 1980.
- [58] TORCZON, V. On the convergence of pattern search algorithms. *SIAM Journal on Optimization* 7, 1 (1997), 1–25.
- [59] TORCZON, V., AND TROSSET, M. W. Using approximations to accelerate engineering design optimization. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (September 1998), pp. 738–748. AIAA paper 98-4800.
- [60] TROSSET, M. W. On the use of direct search methods for stochastic optimization. Tech. Rep. TR00-20, Department of Computational and Applied Mathematics, Rice University, Houston TX, June 2000.
- [61] VAN BEERS, W. C. M., AND KLEIJNEN, J. Kriging interpolation in simulation: A survey. *Proceedings of the 2004 Winter Simulation Conference* (2004), 113–121.
- [62] WATSON, G. S. Smooth regression analysis. *Sankhyā, Series A* 26 (1964), 359–372.
- [63] WOLPERT, D. H., AND MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1 (1997), 67–82.

Index

- Angle Condition, 16
- Backtracking, 17
- Barrier Approach, 30
- Categorical Variables, 5
- Chromosomes, 22
- Convergence, 50
- Cooling Schedule, 19
- Crossover, 23
- Curvature Condition, 16
- Descent Direction, 15
- Deterministic Models, 3
- Discrete Neighbor, 49
- Discrete Plane, 46
- Exact Penalization Approach, 30
- Extended Poll Condition, 53
- Extended Poll Trigger, 53
- Feasible Point Method, 30
- Fitness, 22
- Forcing Function, 37
- Frame, 33
- Fully Sequential Procedure, 59
- Generating Matrix, 31
- Generations, 22
- Improved Mesh Point, 54
- Indifference Parameter, 43
- Individual, 22
- Kernel, 70
- Latin Hypercube Sampling, 73
- Level Set, 27
- Local Minimizer, 49
- Merit Function, 75
- Mesh, 27
- Mesh Local Optimizer, 55
- Mesh Size Parameter, 30
- Mixed Variable Programming, 5
- Mutation, 23
- Nadaraya-Watson Estimator, 69
- Necessary Conditions , 50
- Newton Equation, 13
- Objective Function, 6
- Offspring, 23
- Parents, 23
- Population, 21
- Positive Basis, 47
- Positive Combination, 47
- Positive Spanning Set, 47
- Random Responses, 3
- Search Direction, 12
- Simple Decrease, 14
- Simulated Annealing, 20
- Simulation Models, 1
- Simulation-based Optimization, 2
- Step Size, 12
- Stochastic Models, 3
- Sufficient Decrease, 16
- Sufficient Descent, 15
- Tangent Cone, 51
- Triangular Continuation Region, 60
- Variable Scaling, 76

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 21-03-2005		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) Aug 2004 – Mar 2005	
4. TITLE AND SUBTITLE ON THE USE OF SURROGATE FUNCTIONS FOR MIXED VARIABLE OPTIMIZATION OF SIMULATED SYSTEMS				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Dunlap, John E., 1st Lt, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GOR/ENS/05-06	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Juan R. Vasquez, Major, USAF, Ph.D. Air Force Office of Scientific Research 4015 Wilson Blvd, Room 173 Arlington, VA 22203-1954				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This research considers the efficient numerical solution of linearly constrained mixed variable programming (MVP) problems, in which the objective function is a black-box stochastic simulation, function evaluations may be computationally expensive, and derivative information is typically not available. MVP problems are those with a mixture of continuous, integer, and categorical variables, the latter of which may take on values only from a predefined list and may even be non-numeric. Mixed Variable Generalized Pattern Search with Ranking and Selection (MGPS-RS) is the only existing, provably convergent algorithm that can be applied to this class of problems. Present in this algorithm is an optional framework for constructing and managing less expensive surrogate functions as a means to reduce the number of true function evaluations that are required to find approximate solutions. In this research, the NOMADm software package, an implementation of pattern search for deterministic MVP problems, is modified to incorporate a sequential selection with memory (SSM) ranking and selection procedure for handling stochastic problems. In doing so, the underlying algorithm is modified to make the application of surrogates more efficient. A second class of surrogates based on the Nadaraya-Watson kernel regression estimator is also added to the software. Preliminary computational testing of the modified software is performed to characterize the relative efficiency of selected surrogate functions for mixed variable optimization in simulated systems.					
15. SUBJECT TERMS Mixed variable programming, Pattern Search, Ranking and Selection, Stochastic Optimization, Surrogate Functions, Kernel Regression, Kriging					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)
U	U	U	UU	140	James W. Chrissis, AFIT/ENS (937) 255-3636, ext 4606; e-mail: James.Chrissis@afit.edu